A MARSHALL CAVENDISH 48 COMPUTER COURSE IN WEEKLY PARTS

INPUT

LEARN PROGRAMMING - FOR FUN AND THE FUTURE



UK £1.00 Republic of Ireland £1.25 Malta 85c Australia $2.25 New Zealand $2.95

INPUT

Vol. 4 **No 48**

INDEX

The last part of INPUT, Part 52, will contain a complete, cross-referenced index. For easy access to your growing collection, a cumulative index to the contents of each issue is contained on the inside back cover.

PICTURE CREDITS
Front cover, Graeme Harris. Pages 1443, 1444, 1446, 1449, Stuart Robertson. Pages 1500, 1505, 1507, Graeme Harris. Pages 1508, 1510, Paul Davies. Page 1512, George Logan. Pages 1517, 1518, Dave King. Pages 1520, 1521, 1522, 1524, Peter Richardson.

Published by Marshall Cavendish Partworks Ltd, 58 Old Compton Street, London W1V 5PA, England. Typeset by MS Filmsetting Limited, Frome, Somerset. Printed by Cooper Clegg Web Offset Ltd, Gloucester and Howard Hunt Litho, London.



There are four binders each holding 13 issues.

HOW TO ORDER YOUR BINDERS

UK and Republic of Ireland:
Send £4.95 (inc p & p) (IR£5.95) for each binder to the address below:
Marshall Cavendish Services Ltd, Department 980, Newtown Road, Hove, Sussex BN3 7DN

Australia: See inserts for details, or write to INPUT, Times Consultants, PO Box 213, Alexandria, NSW 2015

New Zealand: See inserts for details, or write to INPUT, Gordon and Gotch (NZ) Ltd, PO Box 1595, Wellington

Malta: Binders are available from local newsagents.

BACK NUMBERS

Back numbers are supplied at the regular cover price (subject to availability).

UK and Republic of Ireland:
INPUT, Dept AN, Marshall Cavendish Services, Newtown Road, Hove BN3 7DN

Australia, New Zealand and Malta:
Back numbers are available through your local newsagent.

COPIES BY POST

Our Subscription Department can supply copies to any UK address regularly at £1.00 each. For example the cost of 26 issues is £26.00; for any other quantity simply multiply the number of issues required by £1.00. Send your order, with payment to:

Subscription Department, Marshall Cavendish Services Ltd, Newtown Road, Hove, Sussex BN3 7DN

Please state the title of the publication and the part from which you wish to start.

HOW TO PAY: Readers in UK and Republic of Ireland: All cheques or postal orders for binders, back numbers and copies by post should be made payable to: *Marshall Cavendish Partworks Ltd.*

QUERIES: When writing in, please give the make and model of your computer, as well as the Part No., page and line where the program is rejected or where it does not work. We can only answer specific queries – and please do not telephone. Send your queries to INPUT Queries, Marshall Cavendish Partworks Ltd, 58 Old Compton Street, London W1V 5PA.

INPUT IS SPECIALLY DESIGNED FOR:

The SINCLAIR ZX SPECTRUM (16K, 48K, 128 and +), COMMODORE 64 and 128, ACORN ELECTRON, BBC B and B+, and the DRAGON 32 and 64.

In addition, many of the programs and explanations are also suitable for the SINCLAIR ZX81, COMMODORE VIC 20, and TANDY COLOUR COMPUTER in 32K with extended BASIC. Programs and text which are specifically for particular machines are indicated by the following symbols:

SPECTRUM 16K, 48K, 128, and +

COMMODORE 64 and 128

ACORN ELECTRON, BBC B and B+

DRAGON 32 and 64

ZX81

VIC 20

TANDY TRS80 COLOUR COMPUTER

ESCAPE: ADDING TO THE ADVENTURE

These program lines conclude the BASIC program. Be sure to SAVE the complete program as it will be used with the compressed text code which follows in the next parts of *INPUT*'s adventure game

```
2830 INPUT "WHAT ARE YOU PREPARED TO
   OFFER?", LINE A$: LET VALUE=Ø
284Ø LET NN=166: GOSUB 45ØØ: IF
   K(21)=-1 AND A$=S$ THEN GOTO
   2890
285Ø LET NN=166: GOSUB 45ØØ: IF
   A$=S$ THEN LET NN=76: GOSUB 396Ø:
   PAUSE 15Ø: RETURN
286Ø DIM G$(17): LET NN=126: GOSUB
   45ØØ: LET G$=S$: DIM B$(17): LET
   NN=77: LET B$=S$: IF A$<>G$ AND
   A$<>B$ THEN GOTO 287Ø
2865 LET NN=7Ø: GOSUB 45ØØ: LET
   G$=S$: IF K(1)=-1 AND E$(L)=G$
   THEN PRINT "IT'S A DEAL": PAUSE 1ØØ:
   LET E$(L)="": LET K(1)=Ø: RETURN
287Ø LET NN=126: GOSUB 45ØØ: IF
   A$<>S$ THEN GOTO 288Ø
2875 DIM G$(17): LET NN=7Ø: GOSUB
   45ØØ: LET G$=S$: IF E$(L)=G$ THEN
   LET NN=78: GOSUB 396Ø: PAUSE 15Ø:
   GOTO 53Ø
288Ø LET NN=166: GOSUB 45ØØ: IF
   A$<>S$ THEN LET NN=79: GOSUB
   396Ø: PRINT A$;"?": PAUSE 15Ø: GOTO
   298Ø
289Ø PRINT "YOU HAVE□";XX;
   "□GOLD SOVEREIGNS.": LET NN=8Ø:
   GOSUB 396Ø: INPUT OFFER
29ØØ IF OFFER>XX THEN LET NN=81:
   GOSUB 396Ø: GOTO 289Ø
291Ø CLS : LET NN=82: GOSUB 396Ø:
   PAUSE 15Ø
292Ø LET PRICE=5Ø*(INT (RND*12)+1)
293Ø IF PRICE>OFFER THEN CLS : LET
   NN=83: GOSUB 396Ø: LET NN=84:
   GOSUB 396Ø: INPUT LINE K$
294Ø IF OFFER>=PRICE THEN GOTO 299Ø
295Ø IF K$="Y" THEN GOTO 289Ø
296Ø LET NN=129: GOSUB 45ØØ: DIM
   G$(17): LET G$=S$: IF K$<>"Y" AND
   E$(L)<>G$ THEN LET NN=85: GOSUB
   396Ø: PAUSE 15Ø: GOSUB 2Ø9Ø: RETURN
297Ø IF K$<>"Y" THEN LET NN=86:
   GOSUB 396Ø: PAUSE 15Ø: PRINT "YOU
   SURRENDER.": PAUSE 5Ø: STOP
298Ø CLS : RETURN
299Ø PRINT "OKAY - IT'S A DEAL": LET
   E$(L)="": LET XX=XX-OFFER
3ØØØ IF XX=Ø THEN LET K(21)=21
3Ø1Ø IF OFFER<>Ø THEN PRINT "YOU'VE
   LOST□";OFFER;"□GOLD SOVEREIGNS"
3Ø2Ø PAUSE 15Ø
3Ø3Ø RETURN
3Ø4Ø REM PROC I
3Ø5Ø CLS
3Ø6Ø PRINT "": LET NN=87: GOSUB 396Ø
3Ø7Ø IF I$=J$ THEN GOTO 325Ø
3Ø8Ø IF I$=U$ AND TT=Ø THEN INPUT
   "WHERE DO YOU WISH TO GO?", LINE
   Q$: LET TT=-1
3Ø9Ø LET NN=88: GOSUB 45ØØ: IF Q$=S$
   THEN LET L=19: RETURN
3Ø95 LET NN=89: GOSUB 45ØØ: IF Q$=S$
   THEN LET L=19: RETURN
31ØØ LET NN=9Ø: GOSUB 45ØØ: IF Q$=S$
   THEN LET L=1: RETURN
31Ø5 LET NN=91: GOSUB 45ØØ: IF Q$=S$
   THEN LET L=1: RETURN
```





```
3110 LET NN=92: GOSUB 45ØØ: IF Q$=S$
   THEN LET L=8: RETURN
3120 LET NN=93: GOSUB 45ØØ: IF Q$=S$
   THEN LET L=2: RETURN
3125 LET NN=94: GOSUB 45ØØ: IF Q$=S$
   THEN LET L=2: RETURN
3130 LET NN=95: GOSUB 45ØØ: IF Q$=S$
   THEN LET L=9: RETURN
314Ø LET NN=96: GOSUB 45ØØ: IF Q$=S$
   THEN LET L=1Ø: RETURN
3145 LET NN=97: GOSUB 45ØØ: IF Q$=S$
   THEN LET L=1Ø: RETURN
315Ø LET NN=98: GOSUB 45ØØ: IF Q$=S$
   THEN LET L=15: RETURN
316Ø LET NN=99: GOSUB 45ØØ: IF Q$=S$
   THEN LET L=21: RETURN
317Ø LET NN=1ØØ: GOSUB 45ØØ: IF
   Q$=S$ THEN LET L=11: RETURN
318Ø LET NN=1Ø1: GOSUB 45ØØ: IF
   Q$=S$ THEN LET L=2Ø: RETURN
3185 LET NN=1Ø2: GOSUB 45ØØ: IF
   Q$=S$ THEN LET L=2Ø: RETURN
319Ø LET NN=1Ø3: GOSUB 45ØØ: IF
   Q$=S$ THEN LET L=17: RETURN
3195 LET NN=1Ø4: GOSUB 45ØØ: IF
   Q$=S$ THEN LET L=17: RETURN
32ØØ LET NN=1Ø5: GOSUB 45ØØ: IF
   Q$=S$ THEN LET L=3: RETURN
32Ø5 LET NN=1Ø6: GOSUB 45ØØ: IF
   A$=S$ THEN LET L=3: RETURN
321Ø LET NN=1Ø7: GOSUB 45ØØ: IF
   Q$=S$ THEN LET L=12: RETURN
322Ø LET NN=1Ø8: GOSUB 45ØØ: IF
   Q$=S$ THEN LET L=13: RETURN
3225 LET NN=1Ø9: GOSUB 45ØØ: IF
   Q$=S$ THEN LET L=13: RETURN
323Ø LET NN=11Ø: GOSUB 45ØØ: IF
   Q$=S$ THEN LET L=5: RETURN
324Ø PRINT '"I DON'T KNOW WHERE
   THE□",Q$;"□IS.": INPUT "TRY
   AGAIN", LINE Q$: GOTO 3Ø9Ø
325Ø LET NN=111: GOSUB 396Ø: PRINT
   E$(L): LET NN=112: GOSUB 396Ø
326Ø DIM G$(17): LET G$=M$: IF
   E$(L)=G$ THEN LET DW=Ø
327Ø LET E$(L)="": LET II=-1
328Ø PAUSE 2ØØ
329Ø RETURN
33ØØ REM PROC J
331Ø LET LL=Ø: FOR Z=1 TO 21
332Ø LET X$=O$(Z): LET Y$=T$: GOSUB
   5ØØØ: IF IN>Ø THEN LET LL=Z
333Ø NEXT Z
334Ø IF LL<1 THEN PRINT "I DON'T
   UNDERSTAND□";T$: GOTO 337Ø
335Ø IF K(LL)<>-1 THEN PRINT "YOU
   HAVEN'T GOT IT!": GOTO 337Ø
336Ø IF K(LL)=-1 THEN PRINT "OKAY -
   YOU'VE LOST IT.": LET K(LL)=L: LET
   PP=PP-1: LET NN=166: GOSUB 45ØØ:
   IF T$=S$ THEN LET XX=Ø
```

```
3370 PAUSE 150
3380 RETURN
3390 REM PROC K
3400 LET NN=146: GOSUB 4500: IF
   T$<>S$ THEN LET NN=202: GOSUB
   3960: PRINT T$;"!": GOTO 3430
3410 IF K(11)<>-1 THEN LET NN=118:
   GOSUB 3960: GOTO 3430
3420 LET NN=203: GOSUB 3960: LET
   NN=204: GOSUB 3960: LET V=V+4:
   LET K(11)=0: LET PP=PP-1
3430 PAUSE 150: RETURN
3440 REM PROC L
3450 LET NN=140: GOSUB 4500: IF
   T$=S$ THEN GOTO 3460
3455 LET NN=162: GOSUB 4500: IF
   T$<>S$ THEN LET NN=113: GOSUB
   3960: PRINT T$;"!": PAUSE 150: RETURN
3460 FOR Z=1 TO 21
3470 DIM G$(17): LET G$=T$: IF
   G$=O$(Z) AND K(Z)<>-1 THEN LET
   NN=118: GOSUB 3960
3480 NEXT Z
3490 LET NN=162: GOSUB 4500: IF T$=S$
   AND K(19)=-1 THEN LET NN=40:
   GOSUB 3960: LET K(19)=0: LET
   V=V+4: LET PP=PP-1
3500 LET NN=140: GOSUB 4500: IF
   T$=S$ AND K(8)=-1 THEN LET
   NN=41: GOSUB 3960: LET K(8)=0: LET
   V=V+10: LET PP=PP-1
3510 PAUSE 150: RETURN
3520 REM PROC M
3530 IF K(2)=-1 THEN LET NN=114:
   GOSUB 3960: GOTO 3550
3540 LET NN=115: GOSUB 3960
3550 PAUSE 250: RETURN
3560 REM PROC N
3570 INPUT "DO YOU WANT TO SAVE
   (Y/N)?", LINE Q$
3580 IF Q$<>"Y" THEN GOTO 3700
3590 SAVE "ADVENTURE" LINE 270
3600 PRINT '"ADVENTURE AND POSITION
   SAVED."
3700 PRINT '"PRESS ANY KEY TO
   CONTINUE. . . .": PAUSE 0
3710 RETURN
3730 REM PROC O
3740 DIM G$(17): LET NN=153: GOSUB
   4500:LET G$=S$: IF E$(L)=G$ THEN
   PRINT "THEY DON'T TAKE TOO KINDLY
   TO□□□THAT. THEY MIGHT GET
   ANGRY.": PAUSE 150: RETURN
3750 LET NN=129: GOSUB 4500: LET
   G$=S$: IF E$(L)<>G$ THEN LET
   NN=116: GOSUB 3960: PRINT E$(L): LET
   NN=117: GOSUB 3960: PAUSE 150:
   RETURN
3760 INPUT "WHAT WITH?",W$
3770 DIM G$(17): LET NN=200: GOSUB
   4500: LET G$=S$: DIM B$(17): LET
   NN=201: GOSUB 4500: LET B$=S$: IF
   W$<>G$ AND W$<>B$ THEN LET
   NN=42: GOSUB 3960: PAUSE 150:
   RETURN
3780 IF K(12)<>-1 THEN LET NN=118:
   GOSUB 3960: PAUSE 150: RETURN
3790 LET NN=43: GOSUB 3960: LET
   E$(L)="": LET K(12)=0: LET
   PP=PP-1
3800 PAUSE 150: RETURN
3810 REM PROC P
3820 CLS : PRINT ': LET NN=119: GOSUB
   3960: LET NN=120: GOSUB 3960
3830 INPUT INVERSE 1;"WHAT NOW?", LINE
   I$
3840 LET NN=121: GOSUB 4500: LET
   Q$=S$: LET NN=122: GOSUB 4500: IF
   I$=Q$ OR I$=S$ THEN GOSUB 1760
3850 LET NN=123: GOSUB 4500: LET
   Q$=S$: LET NN=124: GOSUB 4500: IF
   I$=Q$ OR I$=S$ THEN RETURN
3860 BEEP .2,10: LET NN=125: GOSUB
   3960: GOTO 3830
3870 REM PROC Q
3880 PRINT '"ARE YOU SURE?"
3890 PAUSE 0
3900 IF INKEY$="Y" THEN LOAD
   "ADVENTURE"
3910 RETURN
5020 FOR H=1 TO (LEN X$-LEN Y$+1)
5030 IF Y$=X$(H TO H+LEN Y$-1) THEN
   LET IN=H: LET H=(LEN X$-LEN
   Y$-1)
5040 NEXT H
5050 RETURN
```

```
3130 N=0:S=0:E=0:W=0:U=0:D=0:
   F=1
3140 RETURN
3160 IF E$(L)="" THEN TX=75:GOSUB
   9900:GOSUB 20000:GOTO 3550
3170 TX=153:GOSUB 9950:D1$=Z$:
   TX=155:GOSUB 9950
3175 IF E$(L)=D1$ OR E$(L)=Z$ THEN
   3190
3180 GOTO 3200
3190 PRINT "□O DEAL!":TX=34:GOSUB
   9900:GOSUB 20000:GOTO 10000
3200 PRINT "□HAT ARE YOU PREPARED
   TO OFFER":INPUT OF$:VA=0
3220 TX=166:GOSUB 9950:IF K(21)=-1
   AND OF$=Z$ THEN 3340
3230 TX=166:GOSUB 9950
3235 IF OF$=Z$ THEN TX=76:GOSUB9900:
   GOTO20000
3240 TX=126:GOSUB9950:D1$=Z$:
   TX=77:GOSUB9950
3245 IF OF$=D1$ OR OF$=Z$ THEN 3260
3250 GOTO 3280
3260 TX=70:GOSUB 9900:IF K(1)=-1
   AND E$(L)=Z$ THEN PRINT "□T'S A
   DEAL!"
3270 FOR DL=1 TO 1500:NEXT DL:E$(L)=
   "":K(1)=0:RETURN
3280 IF OF$=D1$ AND E$(L)=Z$ THEN
   3300
3290 GOTO 3310
3300 TX=78:GOSUB 9900:GOSUB 20000:
   GOTO 610
3310 TX=166:GOSUB 9950
3315 IF OF$<>Z$ THEN TX=79:GOSUB
   9950:PRINT Z$OF$"?":GOSUB 20000:
   GOTO 3480
3320 PRINT "□OU HAVE"X"GOLD
   SOVEREIGNS."
3340 TX=80:GOSUB9900
3350 INPUT OF
3360 IF OF>XX THEN TX=81:GOSUB 9900:
   GOTO 3340
3370 PRINT"□":TX=82:GOSUB9900:
   GOSUB 20000
3390 PR=(INT(RND(1)*12)+1)*50
3400 IF PR>OF PRINT "□":PRINT:TX=83:
   GOSUB9900:TX=84:GOSUB9900:INPUT
   IN$
3410 IF OF=>PR THEN 3500
3420 IF IN$="Y" THEN 3340
3430 TX=129:GOSUB 9950
3435 IFIN$<>"Y"ANDE$(L)<>Z$THEN
   TX=85:GOSUB9900:GOSUB20000:GOTO
   2320
3470 IFIN$<>"Y"THENTX=86:GOSUB
   9900:GOSUB20000:PRINT "□OU
   SURRENDER":GOTO10000
3480 PRINT"□":RETURN
3500 PRINT "□KAY-IT'S A DEAL.":
   E$(L)="":XX=XX-OF
3510 IF XX=0 THEN K(21)=21
3520 IF OF<>0 THEN PRINT "□OU'VE
   LOST"OF"GOLD SOVEREIGNS."
3530 GOTO20000
3560 PRINT "□□":TX=87:GOSUB 9900
3680 IF I$=II$ THEN 3880
3690 IF I$=TT$ AND TT=0 THEN INPUT
   "□HERE DO YOU WANT TO GO";DD$:
   TT=-1
3710 TX=88:GOSUB9910:IF DD$=D1$OR
   DD$=Z$THENL=19:RETURN
3720 TX=90:GOSUB9910:IF DD$=D1$OR
   DD$=Z$THENL=1:RETURN
3730 TX=92:GOSUB9950:IF DD$=Z$THEN
   L=8:RETURN
3740 TX=93:GOSUB9910:IF DD$=D1$OR
   DD$=Z$THENL=2:RETURN
3750 TX=95:GOSUB9950:IF DD$=Z$THEN
   L=9:RETURN
3760 TX=96:GOSUB9910:IF DD$=D1$OR
   DD$=Z$THENL=10:RETURN
3770 TX=98:GOSUB9950:IF DD$=Z$THEN
   L=15:RETURN
3780 TX=99:GOSUB9950:IF DD$=Z$THEN
```



```
L=21:RETURN
3790 TX=100:GOSUB9950:IF DD$=Z$
   THENL=11:RETURN
3800 TX=101:GOSUB9910:IF DD$=D1$OR
   DD$=Z$THENL=20:RETURN
3810 TX=103:GOSUB9910:IF DD$=D1$OR
   DD$=Z$THENL=17:RETURN
3820 TX=105:GOSUB9910:IF DD$=D1$OR
   DD$=Z$THENL=3:RETURN
3830 TX=107:GOSUB9950:IF DD$=Z$THEN
   L=12:RETURN
3840 TX=108:GOSUB9910:IF DD$=D1$OR
   DD$=Z$THENL=13:RETURN
3845 TX=110:GOSUB9950:IF DD$=Z$THEN
   L=5:RETURN
3850 PRINT "◩ DON'T KNOW WHERE THE
   "DD$" IS."
3860 PRINT "◫RY AGAIN."
3870 INPUT DD$:GOTO 3710
3880 TX=111:GOSUB9950:PRINTZ$;:
   TX=112:GOSUB9950:PRINT E$(L)Z$
3890 IF E$(L)=JM$ THEN DW=Ø
3900 E$(L)="":II=-1
3910 GOTO 20000
3940LL=Ø:FOR CC=1 TO 21:FOR SC=1
   TO LEN(Ø$(CC))-LEN(T$)+1
3960 IF MID$(Ø$(CC),SC,LEN(T$))=T$
   THEN LL=CC:GOTO 3980
3970 NEXT SC,CC
3980 IF LL<1 THEN PRINT "◩ DON'T
   UNDERSTAND "T$".":GOTO 4030
3990 IF K(LL)<>-1 THEN PRINT "◫OU
   HAVEN'T GOT IT!":GOTO 4030
4000 IF K(LL)=-1 THEN PRINT "◻KAY
   - YOU'VE LOST IT."
4005 IF K(LL)=-1 THEN K(LL)=L:PP=
   PP-1:TX=166:GOSUB 9950:IFT$=Z$
   THENXX=Ø
4030 GOTO 20000
4060 TX=146:GOSUB9950:D1$=Z$:
   TX=202:GOSUB9950:IFT$<>D1$THEN
   PRINTZ$T$"!":GOTO4090
4070 IF K(11)<>-1 THEN TX=118:
   GOSUB9900:GOTO 4090
4080 TX=203:GOSUB 9900:PRINT:TX=204:
   GOSUB 9900:V=V+4:K(11)=Ø:PP=
   PP-1
4090 GOTO 20000
4110 TX=140:GOSUB9950:D1$=Z$:
   TX=162:GOSUB9950:IFT$<>D1$AND
   T$<>Z$THEN4130
4120 GOTO 4140
4130 TX=113:GOSUB9950:PRINT Z$T$"!":
   GOSUB 20000:RETURN
4140 FOR CC=1 TO 21
4150 IF T$=Ø$(CC) AND K(CC)<>-1
   THEN TX=118:GOSUB 9900
4160 NEXT CC
4170 TX=162:GOSUB9950
4175 IFT$=Z$ANDK(19)=-1THENTX=40:
   GOSUB9900:K(19)=Ø:V=V+4:
   PP=PP-1
4180 TX=140:GOSUB9950
4185 IFT$=Z$ANDK(8)=-1THENTX=41:
   GOSUB9900:K(8)=Ø:V=V+10:
   PP=PP-1
4190 GOTO 20000
4210 IF K(2)=-1 THEN TX=114:GOSUB
   9900:GOTO 4230
4220 TX=115:GOSUB 9900
4230 GOTO 20000
4250 PRINT "DO YOU WISH TO SAVE THIS
   GAME - Y/N":INPUT QQ$
4270 IF QQ$<>"Y" THEN 4390
4280 PRINT "PRESS SPACE BAR WHEN
   SAVE TAPE IS READY"
4290 GET G$:IF G$<>"□"
   THEN 4290
4300 DK$=CHR$(13)
4310 INPUT "NAME";N$:OPEN 1,8,1,N$+
   ",S,W"
4320 PRINT#1,L,DK$,BB,DK$,V,DK$,DW,
   DK$,D$,DK$,M$,DK$,
   II$,DK$,TT$
4330 PRINT#1,X,DK$,Q,DK$,KK,DK$,SS,
   DK$,C,DK$,M,DK$,
   XX,DK$,J,DK$,G
4335 PRINT#1,TT,DK$,II,DK$,VV,DK$,F,DK$,
   KK,DK$,QQ,DK$,PP,DK$,OP
4340 FOR CC=1 TO 21
4350 PRINT#1,O$(CC),DK$,E$(CC),DK$,
   K(CC),DK$,F(CC)
4360 NEXT CC:CLOSE1
4380 PRINT"YOUR CURRENT GAME IS
   SAVED."
4390 PRINT"TOUCH SPACE BAR TO
   CONTINUE"
4400 GET GZ$:IF GZ$<>"□" THEN 4400
4410 RETURN
4440 TX=153:GOSUB9950:IF E$(L)=Z$
   THEN 4460
4450 GOTO 4480
4460 PRINT "◫ HEY DON'T TAKE TOO
   KINDLY TO THAT AND MAY GET
   ANGRY!":GOTO20000
4480 TX=129:GOSUB 9950
4485 IF E$(L)<>Z$ THEN TX=116:GOSUB
   9910:PRINT D1$E$(L)Z$:GOTO 20000
4510 PRINT "π○ITH WHAT♠":INPUT
   WW$:PRINT"♰"
4520 TX=200:GOSUB9910:IFWW$<>D1$
   ANDWW$<>Z$THEN4540
4530 GOTO 4550
4540 TX=42:GOSUB 9900:
   GOTO20000
4550 IF K(12)<>-1 THEN TX=118:
   GOSUB 9900:GOTO20000
4560 TX=43:GOSUB 9900:E$(L)="":
   K(12)=Ø:PP=PP-1
4570 GOTO20000
4590 PRINT "♡"
4600 TX=119:GOSUB 9900:PRINT:TX=120:
   GOSUB 9900
4610 PRINT "π○HAT NOW♠":INPUT I$:
   PRINT"♰"
4620 TX=121:GOSUB9910:IF I$=D1$ OR
   I$=Z$ THEN GOSUB 1970
4630 TX=123:GOSUB9910:IF I$=D1$ OR
   I$=Z$ THEN RETURN
4640 TX=125:GOSUB 9900:GOTO 4610
4650 :
4660 PRINT "♡>▌▚INSERT DISK
   ▚▚▚"
4670 INPUT "NAME";N$:OPEN 1,8,Ø,N$+
   ",S,R"
4680 INPUT#1,L,BB,V,DW,D$,M$,II$,TT$,X,
   Q,KK,SS,C,M,XX,J,G,TT,II,VV,F,KK,QQ
4690 INPUT#1,PP,OP:FOR CC=1 TO 21
```



```
4710 INPUT#1,O$(CC),E$(CC),K(CC),F(CC)
4720 NEXT CC:CLOSE1
4740 PRINT "YOUR GAME HAS BEEN
    RESTORED."
4750 TX=70:GOSUB 9900
4755 RESTORE:FOR CC=1 TO 42:READ D1$:
    NEXT
4760 FOR CC=1 TO 32:TX=167+CC:
    GOSUB9950:READ R(CC):R$(CC)=Z$:
    NEXT:RETURN
4765 DATA 1,0,2,0,3,0,0,0,0,4,8,0,9,0,8,0,9,
    0,10,0,11,2,12,0,0,0,14,4
4766 DATA 15,6,0,0,17,0,0,0,19,0,20,1,21,0
4770 DATA 8,5,5,4,8,9,10,9,10,11
4780 DATA 2,2,12,3,3,1,1,1,1,1,1,1
4790 DATA 6,7,12,12,1,1,1,1,1,1
9900 Z$=ZZ$:Z%(0)=A%(TX):SYS 53008:
    PRINT Z$:RETURN
9910 GOSUB 9950:D1$=Z$:TX=TX+1:
    GOSUB 9950:RETURN
9950 Z$=ZZ$:Z%(0)=A%(TX):SYS 53008
9960 Z$=LEFT$(Z$,17)
9970 IFZ$=""THEN RETURN
9980 Z$=LEFT$(Z$,LEN(Z$)-1):IFRIGHT$
    (Z$,1)=CHR$(0)THEN 9970
9990 RETURN
10000 PRINT"■■■PRESS 'Y' FOR
    ANOTHER GAME"
10010 GET A$:IF A$<>"Y" THEN 10010
10020 GOTO 150
20000 FOR Z=1 TO 600:IF PEEK(197)=64
    THEN NEXT Z
20010 RETURN
```

```
3250 PRINTFNW(111)E$(L)FNW(112)
3260 IF E$(L)=JM$ THEN dw=0
3270 E$(L)="":i=-1
3280 D=INKEY(400)
3290 ENDPROC
3300 DEFPROCJ
3310 I=0:FOR c=1 TO 21
3320 IF INSTR(O$(c),T$)>0 THEN I=c
3330 NEXT
3340 IF I<1 THEN PRINT"I don't
    understand□" T$".":GOTO 3370
3350 IF K(I)<>-1 THEN PRINT"You
    haven't got it!":GOTO 3370
3360 PRINT"Okay - you've lost
    it.":K(I)=L:p=p-1: IF
    T$=FNX(FNW(166))THEN x=0
3370 D=INKEY(300)
3380 ENDPROC
3390 DEFPROCK
3400 IF T$<>FNX(FNW(146)) THEN PRINT
    FNW(202)T$"!":GOTO3430
3410 IFK(11)<>-1THEN PRINTFNW(118):
    GOTO3430
3420 PRINTFNW(203) 'FNW(204):V=V+4:
    K(11)=0:p=p-1
3430 D=INKEY(300):ENDPROC
3440 DEFPROCL
3450 IF T$<>FNX(FNW(140)) AND T$
    <>FNX(FNW(162))THEN PRINTFNW
    (113)T$"!":D=INKEY(300):ENDPROC
3460 FOR c=1 TO 21
3470 IF T$=O$(c)AND K(c)<>-1 THEN
    PRINTFNW(118)
3480 NEXT
3490 IF T$=FNX(FNW(162))AND
    K(19)=-1 THENPRINTFNW(40):
    K(19)=0:V=V+4:p=p-1
3500 IF T$=FNX(FNW(140))AND
    K(8)=-1 THEN PRINTFNW(41):K(8)=0:
    V=V+10:p=p-1
3510 D=INKEY(300):ENDPROC
3520 DEFPROCM
3530 IFK(2)=-1THEN PRINTFNW(114):
    GOTO3550
3540 PRINTFNW(115)
3550 D=INKEY(500):ENDPROC
3560 DEFPROCN
3570 INPUT"DO YOU WANT TO SAVE THIS
    PORTION?(y/n)"q$
3580 IF q$<>"y" THEN3700
3590 INPUT"DO YOU KNOW HOW TO SAVE
    A FILE?(y/n)"q$
3600 IF q$<>"y" THEN PRINT"IF USING A
    TAPE RECORDER SET YOUR""RECORDER
    TO RECORD.""USE A BLANK TAPE""DO
    NOT ERASE THE MAIN PROGRAM!""A
    DISK DRIVE WILL TAKE CARE OF ITSELF"
3610 PRINT"PRESS ANY KEY WHEN YOU ARE
    READY"
3620 D$=GET$
3630 O=OPENOUT "Pos"
3640 PRINT#O ,L,b,V,dw,D$,M$,i$,t$,X,Q,k,
    s,C,M,x,J,G,t,i,v,F,KK,qq,p,OP
3650 FOR c=1 TO 21
3660 PRINT#O ,O$ (c) ,E$(c) ,K(c) ,f(c)
3670 NEXT
3680 CLOSE #O
3690 PRINT"YOUR POSITION IS SAVED."
3700 INPUT "PRESS RETURN" GO$
3710 ENDPROC
3730 DEFPROCO
3740 IF E$(L)=FNX(FNW(153)) THENPRINT
    "They don't take too kindly to that. They
    might get angry":D=INKEY(300):
    ENDPROC
3750 IF E$(L)<>FNX(FNW(129)) THEN
    PRINTFNW(116)E$(L)FNW(117):D=INKEY
    (300):ENDPROC
3760 INPUT"What with ",w$
3770 IF w$<>FNX(FNW(200))AND
    w$<>FNX(FNW(201))THEN PRINTFNW
    (42):D=INKEY(300):ENDPROC
3780 IFK(12)<>-1 THEN PRINTFNW
    (118):D=INKEY(300):ENDPROC
3790 PRINTFNW(43):E$(L)="":K(12)=0:
    p=p-1
3800 D=INKEY(300):ENDPROC
3810 DEFPROCP
3820 CLS:PRINT''FNW(119)'FNW(120)
3830 INPUT"What now "I$
3840 IFI$=FNX(FNW(121))ORI$=FNX(FNW
    (122))THENPROCA
3850 IFI$=FNX(FNW(123))ORI$=FNX(FNW
    (124))THENENDPROC
3860 VDU7:PRINTFNW(125):GOTO3830
3870 DEFPROCQ
3880 PRINT"IF USING A RECORDER PRESS
    PLAY"
3890 O=OPENIN "Pos"
3900 INPUT#O ,L,b,V,dw,D$,M$,i$,t$,X,Q,
    k,s,C,M,x,J,G,t,i,v,F,KK,qq,p,OP
3910 FOR c=1 TO 21:INPUT#O,O$(c)
    ,E$(c) ,K(c) ,f(c):NEXT:CLOSE#O
```

```
3920 PRINT"YOUR POSITION IS BEING
    RESTORED": JM$=FNX(FNW(7Ø))
393Ø D=INKEY(3ØØ)
394Ø RESTORE 4Ø4Ø
395Ø FOR c=1 TO32:READ R(c):R$(c)=FNX
    (FNW(167+c)):NEXT:ENDPROC
396Ø DEFFNW(N)
397Ø z%(Ø)=a%(N):CALL DSTRING:=z$
398Ø DEFFNX(z1$)
399Ø z1$=LEFT$(z1$,17)
4ØØØ z1$=LEFT$(z1$,LENz1$-1):IF
    ASCRIGHT$(z1$,1)=Ø THEN4ØØØ
4Ø1Ø =z1$
4Ø2Ø DATA 1,Ø,2,Ø,3,Ø,Ø,Ø,Ø,4,8,Ø,KK,Ø,8,Ø,
    9,Ø,1Ø,Ø,11,2,12,Ø,Ø,Ø,14,4
4Ø3Ø DATA 15,6,Ø,Ø,17,Ø,Ø,Ø,19,Ø,2Ø,1,21,Ø
4Ø4Ø DATA 8,5,5,4,8,9,1Ø,9,1Ø,11
4Ø5Ø DATA 2,2,12,3,3,1,1,1,1,1,1,1
4Ø6Ø DATA 6,7,12,12,1,1,1,1,1,1
```

Tandy owners should make three changes to the program:
In Line 362Ø, change 41194 to 36Ø38.
In Lines 6Ø3Ø and 6Ø4Ø, change Ø6 to 6, and Ø7 to 7.

```
325Ø WN=111: GOSUB5ØØØ:PRINTZ$;:
    WN=112:GOSUB5ØØØ:PRINTE$(L);Z$
326Ø IF E$(L)=JM$ THEN DW=Ø
327Ø E$(L)="":I7=-1
328Ø GOSUB55ØØ
329Ø RETURN
33ØØ REM *** Proc j
331Ø L7=Ø:FOR C7=1 TO 21
332Ø IF INSTR(O$(C7),T$)>Ø THEN L7=C7
333Ø NEXT
334Ø IF L7<1 THEN PRINT"I DON'T
    UNDERSTAND□";T$:GOTO337Ø
335Ø IF K(L7)<>-1 THEN PRINT"YOU
    HAVEN'T GOT IT!":GOTO337Ø
336Ø IF K(L7)=-1 THEN PRINT"OK -
    YOU'VE LOST IT":K(L7)=L:P7=P7-1:
    WN=166:GOSUB52ØØ:IF T$=Z$ THEN
    X7=Ø
337Ø GOSUB55ØØ
338Ø RETURN
339Ø REM***Proc k
34ØØ WN=146:GOSUB52ØØ:IF T$<>Z$
    THEN WN=2Ø2:GOSUB5ØØØ:PRINTZ$;
    "!":GOTO 343Ø
341Ø IF K(11)=-1 THEN WN=118:GOSUB
    51ØØ:GOTO343Ø
342Ø WN=2Ø3:GOSUB51ØØ:WN=2Ø4:
    GOSUB51ØØ:V=V+4:K(11)=Ø:P7=P7
    -1
343Ø GOSUB55ØØ:RETURN
344Ø REM *** Proc l
345Ø WN=14Ø:GOSUB52ØØ:D1$=Z$:WN=
    162:GOSUB52ØØ:IF T$<>D1$ AND
    T$<>Z$ THEN WN=113:GOSUB
    5ØØØ:PRINTZ$;T$;"!":GOSUB55ØØ:
    RETURN
346Ø FORC7=1TO 21
347Ø IF T$=O$(C7)AND K(C7)<>-1
    THEN WN=118:GOSUB51ØØ
348Ø NEXT
349Ø WN=162:GOSUB52ØØ:IF T$=Z$ AND
    K(19)=-1 THEN WN=4Ø:GOSUB51ØØ:
    K(19)=Ø:V=V+4:P7=P7-1
35ØØ WN=14Ø:GOSUB52ØØ:IF T$=Z$
    AND K(8)=-1 THEN WN=41:GOSUB
    51ØØ:K(8)=Ø:V=V+1Ø:P7=P7-1
351Ø GOSUB55ØØ:RETURN
352Ø REM *** Proc m
353Ø IF K(2)=-1 THEN WN=114:GOSUB
    51ØØ:GOTO355Ø
354Ø WN=115:GOSUB51ØØ
355Ø GOSUB55ØØ:RETURN
356Ø REM *** Proc n
357Ø INPUT "SAVE THIS POSITION
    (Y/N)";Q7$
358Ø IF Q7$<>"Y" THEN 37ØØ
359Ø INPUT " DO YOU KNOW HOW TO SAVE
    A FILE (Y/N)";Q7$
36ØØ IF Q7$<>"Y" THEN PRINT"SET UP
    RECORDER TO RECORD. USE A BLANK
    TAPE - TAKE CARE NOT
    TO□□□ERASE THE MAIN PROGRAM!"
361Ø PRINT"PRESS ANY KEY WHEN READY"
362Ø EXEC41194
363Ø OPEN "O",#-1,"POS"
364Ø PRINT#-1,L,B7,V,DW,D$,M$,I7$,
    T7$,X,Q,K7,S7,C,M,X7,J,G,T7,I7,V7,F,KK,
    QQ,P7,OP
365Ø FOR C7=1 TO 21
366Ø PRINT#-1,O$(C7),E$(C7),K(C7),
    F(C7):NEXT
368Ø CLOSE#-1
369Ø PRINT"YOUR POSITION IS SAVED"
37ØØ INPUT "TYPE 'CONT' TO
    CONTINUE";GG$
371Ø IF GG$="CONT" THEN RETURN
372Ø END
373Ø REM *** Proc o
374Ø WN=153:GOSUB52ØØ:IF E$(L)=Z$
    THEN PRINT"THEY DON'T TAKE TOO
    KINDLY TO□□□THAT. THEY MIGHT
    GET ANGRY":GOSUB55ØØ:RETURN
375Ø WN=129:GOSUB52ØØ:IF E$(L)
    <>Z$ THEN WN=116:GOSUB54ØØ:
    PRINTD1$,E(L);Z$:GOSUB55ØØ:RETURN
376Ø INPUT "WITH WHAT";W7$
377Ø WN=2ØØ:GOSUB53ØØ:IF W7$<>
    D1$ AND W7$<>Z$ THEN WN=42:
    GOSUB51ØØ:GOSUB55ØØ:RETURN
378Ø IF K(12)<>-1 THEN WN=118:
    GOSUB51ØØ:GOSUB55ØØ:RETURN
379Ø WN=43:GOSUB51ØØ:E$(L)="":
    K(12)=Ø:P7=P7-1
38ØØ GOSUB55ØØ:RETURN
381Ø REM *** Proc p
382Ø CLS:WN=119:GOSUB51ØØ:WN=12Ø:
    GOSUB51ØØ
383Ø INPUT "WHAT NOW";I$
384Ø WN=121:GOSUB53ØØ:IF I$=D1$ OR
    I$=Z$ THEN GOSUB 176Ø
385Ø WN=123:GOSUB53ØØ:IF I$=D1$ OR
    I$=Z$ THEN RETURN
386Ø WN=125:GOSUB51ØØ:GOTO383Ø
387Ø REM *** Proc q
388Ø PRINT"PRESS play ON CASSETTE"
389Ø OPEN "I",#-1,"POS"
39ØØ INPUT #-1,L,B7,V,DW,D$,M$,I7$,
    T7$,X,Q,K7,S7,C,M,X7,J,G,T7,I7,V7,F,KK,
    QQ,P7,OP
391Ø FOR C7=1 TO 21:INPUT#-1,O$
    (C7),E$(C7),K(C7),F(C7):NEXT:CLOSE
    #-1
392Ø PRINT"YOUR POSITION HAS BEEN
    RESTORED"
393Ø WN=7Ø:GOSUB52ØØ:JM$=Z$
394Ø RESTORE:FOR C7=1 TO 42:READ D1$:
    NEXT
395Ø FOR C7=1 TO 32:WN=167+C7:
    GOSUB52ØØ:READ R(C7):R$(C7)=Z$:
    NEXT:RETURN
4Ø2Ø DATA 1,Ø,2,Ø,3,Ø,Ø,Ø,Ø,4,8,Ø,9,Ø,8,Ø,9,
    Ø,1Ø,Ø,11,2,12,Ø,Ø,Ø,14,4
4Ø3Ø DATA 15,6,Ø,Ø,17,Ø,Ø,Ø,19,Ø,2Ø,1,21,Ø
4Ø4Ø DATA 8,5,5,4,8,9,1Ø,9,1Ø,11
4Ø5Ø DATA 2,2,12,3,3,1,1,1,1,1,1,1
4Ø6Ø DATA 6,7,12,12,1,1,1,1,1,1
5ØØØ REM *** fnw >WN <Z$,ZO$
5Ø1Ø ZO$=Z1$:XX=FNW(WN)
5Ø15 IF ASC(ZO$)=Ø THEN Z$="":
    RETURN
5Ø2Ø Z$=ZO$:RETURN
51ØØ REM *** PRINT fnw >WN <Z$
511Ø GOSUB 5ØØØ:PRINTZ$:RETURN
52ØØ REM *** fnx >WN <Z$
521Ø GOSUB 5ØØØ
522Ø Z$=LEFT$(Z$,17)
523Ø Z$=LEFT$(Z$,LEN(Z$)-1):IF ASC
    (RIGHT$(Z$,1))=Ø THEN 523Ø
524Ø RETURN
53ØØ GOSUB 52ØØ:D1$=Z$:WN=WN+1:
    GOSUB52ØØ:RETURN
54ØØ GOSUB 5ØØØ:D1$=Z$:WN=WN+1:
    GOSUB5ØØØ:RETURN
55ØØ FOR DU=1 TO 6ØØ:IF INKEY$=""
    THEN NEXT
551Ø RETURN
6ØØØ DEF USR6=325Ø7
6ØØ5 DEF USR7=32496
6Ø15 XX=RND(-TIMER)
6Ø3Ø DEFFNW(N)=USRØ6(3Ø77+A(N))
6Ø4Ø XX=USRØ7(VARPTR(ZO$))
6Ø5Ø RETURN
65ØØ PRINT:PRINT:PRINT"PRESS 'Y' FOR
    ANOTHER GAME"
651Ø IFINKEY$<>"Y" THEN 651Ø
652Ø GOTO11Ø
```



PAPER, SCISSORS, STONE

So you think you can out-bluff a mere machine? Try these two programs and find out otherwise. Your computer uses statistics to capitalise on your biases



- BLUFFING GAMES
- RANDOMNESS
- OUTCOMES
- SCISSORS, PAPER, STONE GAME
- USING STATISTICS
- EXPONENTIAL SMOOTHING TECHNIQUE
- CUMULATIVE SUM TECHNIQUE



The computer that can 'out-bluff' a human isn't a beast of the future, it's here right now. With a BASIC program, your computer will almost certainly be able to beat you and your friends when playing bluffing games.

IT'S A SNIP

Bluffing games are games in which both participants' moves are made at the same time, and only revealed afterwards. The only information open to either side is 'how your opponent thinks'.

A classic example of a bluffing game is Scissors, Paper, Stone, in which two players each make the shape of one of the three alternatives with one hand. They simultaneously show each other what they have chosen. The winner is the player who is showing the most powerful of the two objects which are shown, according to these rules:

- scissors cut paper, so scissors win
- paper wraps stone, so paper wins
- stone blunts scissors, so stone wins

If both players choose the same object, it's a draw.

In this article you'll see how to write two different versions of a program which will play a very good game of Scissors, Paper, Stone. And you may find playing these games a little unsettling, as your computer seems to be able to anticipate your thoughts.

MORE ABOUT BLUFFING

How can you write a program which anticipates seemingly random events? One way of doing this would be to make the outcome pure chance. You could simply write a program which picks numbers at random to play against the responses of the human player. But such a game would be very unsatisfactory. If the random number generator is totally random, the best outcome you could possibly hope for your program would probably be a draw (if you took enough turns). But in practice, the actual outcome would be affected by one other crucial factor—the random number generator in your machine isn't perfectly random. It's biased.

This would make such a game even more unsatisfactory, because if you could discover the machine's bias, you could beat it simply by aiming for better than the number (or corresponding symbol) the machine tends to produce. The process of discovering the bias involves a number of statistical calculations.

Although you might not be very good at performing lightning-fast statistical calculation, in order to beat an opponent, you could use a computer to do the dirty work. And this gives a clue as to how you could program a computer to play against a human.

YOU'RE BIASED!

Unlike the computer's bias, a human's bias is not constant. Instead of tending towards the same response all through play, the human's pattern is more complex.

Aside from humans' failings at generating random numbers, the human player will also respond to what's happening in the game, almost certainly formulating theories of which responses are more likely to win, or to lose. The distractions of playing the game also reduce the human player's capability of producing truly random responses.

Another way of saying all this is that a human player will almost certainly actively seek to maximize his or her returns (win!). In doing so, the player introduces some kind of bias, which with careful programming, can be capitalized upon by the computer.

If the player tends to change his strategy gradually, then you should use a statistical technique known as *exponential smoothing*. Conversely, if the opponent changes his tactics quickly, then the program should employ *cumulative sum* techniques. Unfortunately, it's just pot luck when it comes to choosing which program you should use to frustrate your friends.

PLAY THE GAME

Playing Scissors, Paper, Stone on a computer is just as simple as playing your friends. Instead of making shapes behind your back, you press either the 1, 2 or 3 key.

It's important to realise that, even though at least one of the programs is likely to be able to beat you, the computer isn't cheating. The computer doesn't look at your choice before making its choice. How the programs work is to make the choice for the coming round at the

end of each round—the computer has already made its choice before you press one of the keys.

EXPONENTIAL SMOOTHING

Exponential smoothing is a statistical technique in which the importance of old responses is reduced in comparison with recent responses. RUN the program and you will be asked first to INPUT a 'forgetting factor'. The higher the number you give it, the more it forgets and the easier your task is likely to be to beat it.

```
5 CLEAR 31999: GOSUB 5ØØ: BORDER Ø:
   PAPER Ø: INK 7: CLS
7 DIM A$(3,9): LET A$(1)="STONE": LET
   A$(2)="PAPER": LET
   A$(3)="SCISSORS"
1Ø DIM H(3,2): DIM X(2): DIM Q(3,3): DIM
   C(2,3): DIM A(2,3): DIM P(3,3): CLS
2Ø FOR I=1 TO 3: LET H(I,1)=-COS
   ((I-2)*PI*2/3): LET H(I,2)=-SIN
   ((I-2)*PI*2/3): NEXT I
3Ø FOR T=1 TO 3: FOR J=1 TO 2: LET
   A(J,T)=.Ø1: NEXT J: NEXT T
4Ø INPUT "ENTER FORGETTING FACTOR TO
   BE□□□USED,□(Ø-1).□
   SUGGESTED VALUE□□□□Ø.85□";W
7Ø FOR I=1 TO 3: FOR J=1 TO 3: LET
   P(I,J)=SGN (I-J-3*INT ((I-J+1.5)
   /3)): NEXT J: NEXT I: LET S=Ø
8Ø LET U=1: LET S=Ø: LET U2=Ø: LET
   WW=Ø: LET U3=Ø
1ØØ LET V=INT (RND*3)+1
11Ø PRINT INK 6; PAPER 2;"□□1=STONE,
   □2=PAPER,□3=SCISSORS□□"
12Ø INK 6: PLOT 8,167: DRAW 239,Ø: DRAW
   Ø,-159: DRAW -239,Ø: DRAW Ø,159
2ØØ FOR T=1 TO 3: LET I=U
21Ø IF (U2=Ø AND T=2) OR (U3=Ø AND
   T=3) THEN GOTO 28Ø
22Ø IF T=2 THEN LET I=ABS
   ((U=U2)-2*(U=VV)-3*((U<>U2)
   AND (U<>VV)))
23Ø IF T=3 THEN LET I=ABS
   ((U=U3)-2*(U=V3)-3*((U<>U3)
   AND (U<>V3)))
24Ø FOR J=1 TO 2: LET A(J,T)=A(J,T)*
   W+H(I,J): LET C(J,T)=C(J,T)*W+3*H
   (I,J)*H(I,J)+.Ø1
25Ø LET X(J)=A(J,T)/C(J,T): NEXT J
26Ø FOR I=1 TO 3: LET Q(I,T)=1/3: FOR
   K=1 TO 2: LET Q(I,T)=X(K)*H(I,K): NEXT
   K: NEXT I
28Ø NEXT T: LET U2=U: LET VV=V: IF
   U=V THEN LET VV=U+1-3*INT (U/3)
29Ø IF U<>V THEN LET U3=U: LET
   V3=V: IF P(U3,V3)<Ø THEN LET
   WW=U3: LET U3=V3: LET V3=WW
3ØØ LET X=-1E3Ø: FOR T=1 TO 3: IF
   (T=2 AND U2=Ø) OR (T=3 AND
   U3=Ø) THEN GOTO 37Ø
31Ø IF T=1 THEN GOTO 35Ø
311 IF T=2 THEN GOTO 32Ø
312 IF T=3 THEN GOTO 34Ø
32Ø LET WW=6-U2-VV: LET Q1=Q(1,T):
   LET Q2=Q(2,T): LET Q3=Q(3,T): LET
   Q(U2,T)=Q1: LET Q(VV,T)=Q2: LET
   Q(WW,T)=Q3
33Ø GOTO 35Ø
34Ø LET WW=6-U3-V3: LET Q1=Q(1,T):
   LET Q2=Q(2,T): LET Q3=Q(3,T): LET
   Q(U3,T)=Q1: LET Q(V3,T)=Q2: LET
   Q(WW,T)=Q3
35Ø FOR G=1 TO 3: LET P=Ø: FOR I=1 TO
   3: LET P=P+P(G,I)*Q(I,T): NEXT I: IF
   P>X THEN LET X=P: LET V=G
36Ø NEXT G
37Ø NEXT T
4ØØ INK 7: PRINT AT 2,4; "YOU SAID□"
4Ø5 FOR M=-3 TO 16: BEEP .Ø1,M: NEXT M
41Ø LET K$=INKEY$: IF K$="" THEN
   GOTO 41Ø
412 IF K$<"1" OR K$>"3" THEN GOTO
   41Ø
415 LET U=VAL K$
42Ø PRINT AT 3,4;A$(U);AT 2,21; INK 5;"I
   SAID□";AT 3,21;A$(V)
43Ø POKE 23681,U-1: LET O=USR 32ØØØ:
   POKE 23681,127+V: LET O=USR 32ØØØ
44Ø LET S=S+P(U,V): PRINT AT
   16,8;"YOUR SCORE IS□";S;"□□"
45Ø INVERSE 1: IF V=U THEN PRINT AT
   18,1Ø;"□IT'S A DRAW□": GOTO 49Ø
46Ø IF (U=3 AND V=2) OR (U=2 AND
   V=1) OR (U=1 AND V=3) THEN PRINT
   AT 18,12;"□YOU WIN□": GOTO 49Ø
47Ø PRINT AT 18,11;"□□I WIN□□"
49Ø INVERSE Ø: FOR D=1 TO 2: PAUSE Ø:
   NEXT D: FOR N=2 TO 18: PRINT PAPER
   Ø; INK 7;AT N,2;"□□□□□□□□□□
   □□□□□□□□□□□□□□□□□□
   □□□□": NEXT N: GOTO 2ØØ
5ØØ FOR N=32ØØØ TO 32284: READ A:
   POKE N,A: NEXT N
5Ø5 LET N=32Ø69: POKE 23728,N-256*INT
   (N/256): POKE 23729,INT (N/256)
51Ø RETURN
52Ø DATA 33,6,72,58,129,92,2Ø3,127,4Ø,5,
   2Ø3,191,33,22,72,221,42,176,92,17,72,Ø,
   254,Ø,4Ø
53Ø DATA 8,254,1,4Ø,2,221,25,221,25,221,
   229,2Ø9,6,3,197,229,6,8,197,1,3,Ø,235,
   237,176,235,36,1,3,Ø
54Ø DATA 237,66,193,16,239,225,1,32,Ø,9,
   193,16,227,2Ø1
55Ø DATA Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,3Ø,Ø,Ø,
   97,128,3,129,192,4,15,224,1Ø,56,112,21,
   32,48,3Ø,192,16,11,16Ø,16,7,12Ø,32
56Ø DATA 3,181,96,1,234,192,Ø,255,128,Ø,Ø,
   Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,
   Ø,Ø
57Ø DATA Ø,16,Ø,Ø,56,Ø,Ø,124,Ø,Ø,254,Ø,1,
   223,Ø,3,111,128,7,191,192,14,219,224,27,
   1Ø9,24Ø,53,254,248,1Ø6,219,112,245,1Ø9,
   224,122
58Ø DATA 19Ø,192,61,91,128,3Ø,239,Ø,15,182,
   Ø,7,92,Ø,2,232,Ø,1,24Ø,Ø,Ø,224,Ø,Ø,Ø,Ø
59Ø DATA Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
6ØØ DATA Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,56,Ø,Ø,1Ø4,Ø,Ø,
   2Ø8,Ø,1,16Ø,Ø,3,64,Ø,6,128,Ø,13,Ø,Ø,26,Ø,
   Ø,52,Ø,1,232,Ø,63,48,Ø
61Ø DATA 1ØØ,32,Ø,196,96,Ø,2Ø1,192,Ø,115,
   96,Ø,6,32,Ø,4,32,Ø,6,64,Ø,3,128,Ø,Ø,Ø,Ø,Ø,
   Ø,Ø,Ø,Ø,Ø
```

```
4 FOR Z=12288 TO 12479:READ X:POKE
   Z,X:NEXT Z
5 SP=53248:POKESP,8Ø:POKESP+1,1ØØ:
   POKESP+2,23Ø:POKESP+3,1ØØ
6 POKE SP+23,255:POKE SP+29,255:POKE
   5328Ø,6:POKE53281,3
7 A$(1)="STONE□□□":A$(2)=
"PAPER□□□":A$(3)="SCISSORS"
1Ø DIM H(3,2),X(2),Q(3,3),C(2,3),A(2,3),
   P(3,3)
2Ø FORI=1TO3:H(I,1)=-COS((I-2)*π*2/3)
   :H(I,2)=-SIN((I-2)*π*2/3):NEXTI
3Ø FORT=1TO3:FORJ=1TO2:A(J,T)=Ø:
   C(J,T)=.Ø1:NEXTJ:NEXTT
4Ø PRINT"[CLR][DOWN]INPUT FORGETTING FACTOR
   TO BE USED)):PRINT "PERMITTED RANGE
   Ø TO 1"
5Ø PRINT"[RVS]Ø=NO MEMORY BEYOND ONE
   GAME":PRINT"1=UNCHANGING
   STRATEGY"
6Ø PRINT"[RVS]SUGGESTED VALUE=
   .85[RVS]":INPUTW
7Ø FORI=1TO3:FORJ=1TO3:P(I,J)=SGN
   (I-J-3*INT((I-J+1.5)/3)):NEXTJ:
   NEXTI:S=Ø
8Ø S=Ø:U2=Ø:WW=Ø:U3=Ø
1ØØ V=INT(RND(Ø)*3)+1
11Ø PRINT"[CLR][RVS][HOME]"TAB(7)"1=STONE,
   2=PAPER,3=SCISSORS":POKESP+21,Ø
12Ø PRINT"▙▙◆◪⊟⊟⊟⊟⊟⊟
   ⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟
   ⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟
   ⊟⊟◳"
13Ø FORZ=1TO15:PRINT"▙▙◫□□□
□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□
◫":NEXTZ
14Ø PRINT"▙▙◰⊟⊟⊟⊟⊟⊟⊟
   ⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟
   ⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟⊟
   ⊟◲":GOTO 4ØØ
2ØØ FORT=1TO3:I=U
21Ø IF(U2=ØANDT=2)OR(U3=ØANDT=3)
   THEN28Ø
```

```
220 IFT=2THENI=-(U=U2)-2*(U=
    VV)-3*((U<>U2)AND(U<>VV))
230 IFT=3THENI=-(U=U3)-2*(U=
    V3)-3*((U<>U3)AND(U<>V3))
240 FORJ=1TO2:A(J,T)=A(J,T)*W+H(I,J):
    C(J,T)=C(J,T)*W+3*H(I,J)*H(I,J)+Ø.Ø1
250 X(J)=A(J,T)/C(J,T):NEXTJ
260 FORI=1TO3:Q(I,T)=1/3:FORK=1TO2:
    Q(I,T)=Q(I,T)+X(K)*H(I,K):NEXTK:NEXTI
280 NEXTT:U2=U:VV=V:IFU=VTHEN
    VV=U+1-3*INT(U/3)
290 IFU<>VTHENU3=U:V3=V:IFP(U3,
    V3)<ØTHENWW=U3:U3=V3:V3=WW
300 X=-1E3Ø:FORT=1TO3:IF(T=2AND
    U2=Ø)OR(T=3ANDU3=Ø)THEN37Ø
310 ONTGOTO35Ø,32Ø,34Ø
320 WW=6-U2-VV:Q1=Q(1,T):
    Q2=Q(2,T):Q3=Q(3,T):Q(U2,T)=Q1:
    Q(VV,T)=Q2:Q(WW,T)=Q3
330 GOTO35Ø
340 WW=6-U3-V3:Q1=Q(1,T):
    Q2=Q(2,T):Q3=Q(3,T):Q(U3,T)=Q1:
    Q(V3,T)=Q2:Q(WW,T)=Q3
350 FORG=1TO3:P=Ø:FORI=1TO3
    P=P+P(G,I)*Q(I,T):NEXTI:IFP>X
    THENX=P:V=G
360 NEXTG
370 NEXTT
400 PRINT"■■■■■■■■■
    YOU SAID■?□□□□□□□□□□
    ▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌▌";:
    GOSUB 7ØØ
410 GETA$:U=VAL(A$):IFU<1ORU>3
    THEN41Ø
420 PRINTA$(U)"■■■I SAID■";A$(V)
430 POKE 2Ø4Ø,191+U:POKE2Ø41,191+V:
    POKE SP+21,3
440 S=S+P(U,V):PRINT "■■■■■
    ■■■■■"TAB(14)"■YOUR
    SCORE";S"▌□□■■■■■"
450 PRINT "■";TAB(14);:IF U=V THEN
    PRINT "□IT'S A DRAW□":GOTO 49Ø
460 IF (U=3ANDV=2)OR(U=2ANDV=1)
    OR(U=1ANDV=3)THEN PRINT "□
    □□YOU WIN□□□":GOTO 49Ø
470 PRINT "□□□I WIN□□□"
490 FOR Z=1 TO 999:NEXT Z:PRINT
    "■■"TAB(14);:FOR Z=1 TO 13:
    PRINT "□";:NEXT Z
500 PRINT:POKE198,Ø:GOTO 2ØØ
600 DATA Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,3Ø,Ø,Ø,
    97,128,3,129,192,4,15,224,1Ø,56
610 DATA 112,21,32,48,3Ø,192,16,11,16Ø,16,
    7,12Ø,32,3,181,96,1,234,192,Ø,255
620 DATA 128,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
630 DATA Ø,16,Ø,Ø,56,Ø,Ø,124,Ø,Ø,254,Ø,1,
    223,Ø,3,111,128,7,191,192,14,219
640 DATA 224,27,1Ø9,24Ø,53,254,248,1Ø6,
    219,112,245,1Ø9,224,122,19Ø,192,61
650 DATA 91,128,3Ø,239,Ø,15,182,Ø,7,92,Ø,2,
    232,Ø,1,24Ø,Ø,Ø,224,Ø,Ø,Ø,Ø,Ø
660 DATA Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,56,Ø,Ø,1Ø4,Ø,Ø,
    2Ø8,Ø,1,16Ø,Ø,3,64,Ø,6,128,Ø,13,Ø,Ø,26
670 DATA Ø,Ø,52,Ø,1,232,Ø,63,48,Ø,1ØØ,32,Ø,
    196,96,Ø,2Ø1,192,Ø,115,96,Ø,6,32,Ø
680 DATA 4,32,Ø,6,64,Ø,3,128,Ø,Ø
700 POKE 54277,33:POKE 54278,255:POKE
    54296,15
710 POKE 54276,33:FOR DE=1 TO 5Ø:POKE
    54273,5Ø-DE:NEXT DE:POKE 54296,32:
    RETURN
```

```
10 CLS
20 DIMA$(3):A$(1)="STONE":A$(2)=
   "PAPER":A$(3)="SCISSORS"
30 DIMH(3,2),X(2),Q(3,3),C(2,3),A(2,3),P(3,3)
40 FORT=1TO3:H(T,1)=-COS((T-2)*PI*
   2/3):H(T,2)=-SIN((T-2)*PI*2/3):NEXT
50 FORT=1TO3:FORP=1TO2:A(P,T)=.Ø1:
   NEXT:NEXT
60 INPUT"FORGETTING FACTOR TO BE
   USED(Ø-1)""TRY .85",W:IF W>1
   ORW<ØTHEN6Ø
70 FORT=1TO3:FORP=1TO3:P(T,P)=SGN
   (T-P-3*INT((T-P+1.5)/3)):NEXT:
   NEXT
80 U=1:S=Ø:U2=Ø:WW=Ø:U3=Ø:
   V=RND(3)
90 MODE1
100 PRINTTAB(5,1)"1=STONE□□□□
    2=PAPER□□□□3=SCISSORS"
110 FORT=1TO3:I=U
120 IF(U2=ØANDT=2)OR(U3=ØANDT=3)
    THEN18Ø
130 IFT=2THENI=ABS((U=U2)-2*(U=
    VV)-3*((U<>U2)AND(U<>VV)))
140 IFT=3THENI=ABS((U=U3)-2*(U=
    V3)-3*((U<>U3)AND(U<>V3)))
150 FORJ=1TO2:A(J,T)=A(J,T)*W+H(I,J):
    C(J,T)=C(J,T)*W+3*H(I,J)*H(I,J)+.Ø1
160 X(J)=A(J,T)/C(J,T):NEXT
170 FORI=1TO3:Q(I,T)=1/3:FORK=1TO2:
    Q(I,T)=X(K)*H(I,K):NEXT:NEXT
180 NEXT:U2=U:VV=V:IFU=V□THEN
    VV=U+1-3*INT(U/3)
190 IFU<>V□THENU3=U:V3=V:IFP(U3,
    V3)<ØTHENWW=U3:U3=V3:V3=WW
200 X=-1E3Ø:FORT=1TO3:IF(T=2AND
    U2=Ø)OR(T=3ANDU3=Ø)THEN27Ø
210 ON(T)GOTO25Ø,22Ø,24Ø
220 WW=6-U2-VV:Q1=Q(1,T):
    Q2=Q(2,T):Q3=Q(3,T):Q(U2,T)=Q1:
    Q(VV,T)=Q2:Q(WW,T)=Q3
230 GOTO25Ø
240 WW=6-U3-V3:Q1=Q(1,T):
    Q2=Q(2,T):Q3=Q(3,T):Q(U3,T)=Q1:
    Q(V3,T)=Q2:Q(WW,T)=Q3
250 FORG=1TO3:P=Ø:FORI=1TO3:
    P=P+P(G,I)*Q(I,T):NEXT:IFP>X□
    THENX=P:V=G
260 NEXT
270 NEXT
280 PRINTTAB(4,4)"YOU SAID□";
290 FORM=ØTO2ØSTEP2:SOUND1,-5,M*5,
    1:NEXT
300 K$=GET$
310 IFK$<"1" ORK$>"3" THEN 3ØØ
320 U=VALK$
330 PRINTA$(U)"□□□I SAID□"A$(V)
340 VDU29,3ØØ;512;:PROCOBJ(U):VDU29,
    9ØØ;512;:PROCOBJ(V)
350 S=S+P(U,V):PRINTTAB(12,25)"YOUR
    SCORE IS□";S
360 IF V=U□THEN PRINTTAB(15,22)
    "IT'S A DRAW":GOTO39Ø
370 IF(U=3ANDV=2)OR(U=2ANDV=1)
    OR(U=1ANDV=3)THENPRINTTAB(15,22)
    "YOU WIN":GOTO39Ø
380 PRINTTAB(17,22)"I WIN"
390 D=GET:GOTO9Ø
400 DEFPROCOBJ(JM)
410 ON(JM)GOTO54Ø,42Ø,59Ø
420 R=1Ø:FOR T=-PI*6TOØSTEP.4
430 PLOT69,COS(T)*R,SIN(T)*R/2+2ØØ
440 R=R+.8
450 IFT>-PI□THENPLOT69,COS(T)*R,
    SIN(T)*R/2
460 NEXT
470 FOR T=PI□TO PI/2STEP-.2
480 PLOT69,COS(T)*R+R+R,SIN(T)*
    R/2+2ØØ
490 PLOT69,COS(T)*R+R+R,SIN(T)*R/2
500 NEXT
510 MOVE-R*.9,Ø:DRAW-R*.9,2ØØ
520 MOVER*2,R/2:DRAWR*2,R/2+2ØØ
530 ENDPROC
540 R=1ØØ:MOVE256,64
550 FORT=-PI□TOPI□STEP.2
560 DRAW-COS(T)*R+ABS(T*5Ø),SIN(T)*
    R+ABS(T*2Ø)
570 NEXT
580 ENDPROC
590 R=3Ø:FORT=ØTOPI*2STEP.2
600 PLOT69,COS(T)*R,SIN(T)*R
610 PLOT69,85+COS(T)*R,1ØØ+SIN(T)*R
620 NEXT
630 MOVE-25,1Ø:DRAW-2ØØ,3ØØ:DRAW
    -8,3Ø
640 MOVE6Ø,11Ø:DRAW-3ØØ,19Ø:DRAW
    6Ø,9Ø
650 ENDPROC
```

```
5 CLEAR1ØØØ:PMODE3,1:COLOR4,2:PCLS
6 GOSUB1ØØØ
7 C=8*ATN(1)/3
10 DIMH(3,2),X(2),Q(3,3),C(2,3),A(2,3),
   P(3,3):CLS
20 FORI=1TO3:H(I,1)=-COS((I-2)*C):
   H(I,2)=-SIN((I-2)*C):NEXT
30 FORT=1TO3:FORJ=1TO2:A(J,T)=Ø:
   C(J,T)=.Ø1:NEXTJ,T
```

```
40 PRINT"INPUT FORGETTING FACTOR TO
   BE□□□USED. PERMITTED RANGE Ø
   TO 1"
50 PRINT"Ø=NO MEMORY BEYOND ONE
   GAME":PRINT"1=UNCHANGING
   STRATEGY"
60 PRINT"SUGGESTED VALUE=.85":INPUT W
70 FORI=1TO3:FORJ=1TO3:P(I,J)=SGN
   (I-J-3*INT((I-J+1.5)/3)):NEXTJ,I
80 S=Ø:U2=Ø:WW=Ø:U3=Ø
100 V=RND(3)
110 COLOR4:PCLS:LINE(8,2Ø)-(247,171),
    PSET,B
120 DRAW"BM34,1ØS4C4":FORK=1TO3:
    DRAWN$(K)+E$+A$(K)+"BR4":NEXT
140 GOTO4ØØ
200 FORT=1TO3:I=U
210 IF(U2=ØANDT=2)OR(U3=ØANDT=3)
    THEN28Ø
220 IFT=2THENI=-(U=U2)-2*
    (U=VV)-3*((U<>U2)AND(U<>VV))
230 IFT=3THENI=-(U=U3)-2*
    (U=V3)-3*((U<>U3)AND(U<>V3))
240 FORJ=1TO2:A(J,T)=A(J,T)*W+H(I,J):
    C(J,T)=C(J,T)*W+3*H(I,J)*H(I,J)+.Ø1
250 X(J)=A(J,T)/C(J,T):NEXT
260 FORI=1TO3:Q(I,T)=1/3:FORK=1TO2:
    Q(I,T)=Q(I,T)+X(K)*H(I,K):NEXTK,I
280 NEXT:U2=U:VV=V:IFU=V□THEN
    VV=U+1-3*INT(U/3)
290 IFU<>V THENU3=U:V3=V:IFP(U3,
    V3)<Ø THENWW=U3:U3=V3:V3=WW
300 X=-1E3Ø:FORT=1TO3:IF(T=2AND
    U2=Ø)OR(T=3ANDU3=Ø)THEN37Ø
310 ON T GOTO35Ø,32Ø,34Ø
320 WW=6-U2-VV:Q1=Q(1,T):Q2=Q
    (2,T):Q3=Q(3,T):Q(U2,T)=Q1:Q(VV,T)=
    Q2:Q(WW,T)=Q3
330 GOTO35Ø
340 WW=6-U3-V3:Q1=Q(1,T):Q2=Q
    (2,T):Q3=Q(3,T):Q(U3,T)=Q1:
    Q(V3,T)=Q2:Q(WW,T)=Q3
350 FORG=1TO3:P=Ø:FORI=1TO3:
    P=P+P(G,I)*Q(I,T):NEXT:IFP>X□
    THENX=P:V=G
360 NEXTG
370 NEXTT
400 SCREEN1,Ø:DRAW"BM12,3Ø"+Y$+S$
410 A$=INKEY$:IFA$<"1" ORA$>"3"
    THEN41Ø
420 U=VAL(A$):DRAWA$(U)+
    "BM142,3Ø"+I$+S$+A$(V)
430 X=44:Y=9Ø:ON U GOSUB6ØØ,61Ø,62Ø
435 X=17Ø:ON V GOSUB 7ØØ,71Ø,72Ø
440 LINE(8Ø,175)-(24Ø,19Ø),PRESET,BF:
    S=S+P(U,V):DRAW"BM8Ø,18ØBD4R5U
    2L4U2R4BR8L4D4R4BR4U4R4D4NL4BR4U
    4R4D2L2DFBR9L4U2NRU2R4BR8BU2S8":
    GOSUB8ØØ:DRAW"S4"
450 IF U=V THENDRAW"BM1ØØ,16ØC3"+
    D$:GOTO49Ø
460 IF (U=3ANDV=2)OR(U=2ANDV=1)
    OR(U=1ANDV=3) THENDRAW"BM3Ø,
    16ØC3"+Y$+W$:GOTO49Ø
470 DRAW"BM16Ø,16ØC1"+I$+W$
490 FORK=1TO1ØØØ:NEXT:LINE(1Ø,22)-
    (245,169),PRESET,BF
500 GOTO2ØØ
600 PUT(X,Y)-(X+4Ø,Y+38),PA,PSET:
    RETURN
610 PUT(X,Y)-(X+4Ø,Y+38),SC,PSET:
    RETURN
620 PUT(X,Y)-(X+4Ø,Y+38),ST,PSET:
    RETURN
700 PUT(X,Y)-(X+4Ø,Y+38),PA,AND:
    RETURN
710 PUT(X,Y)-(X+4Ø,Y+38),SC,AND:
    RETURN
720 PUT(X,Y)-(X+4Ø,Y+38),ST,AND:
    RETURN
800 FORK=1TOLEN(STR$(S))
810 B$=MID$(STR$(S),K,1):IFB$="-"
    THENDRAW"BF2R4BE2":GOTO83Ø
820 IFB$<"Ø" OR B$>"9" THEN83Ø
825 DRAWN$(VAL(B$))
830 NEXT:RETURN
1000 FORI=ØTO9:READN$(I):NEXT
1010 DATA NR2D4R2U4BR2,BDEND4BR2,R2
     D2L2D2R2BU4BR2,NR2BD2NR2BD2R2U4
     BR2,D2R2D2U4BR2,NR2D2R2D2L2BE4,D4
     R2U2L2BE2BR2,R2ND4BR2,NR2D4R2U2
     NL2U2BR2,NR2D2R2D2U4BR2
1015 DIMPA(39),SC(39),ST(39)
1020 DRAW"BMØ,22C3M22,ØM4Ø,18M2Ø,
     38L4MØ,22":PAINT(2Ø,2Ø)
1030 DRAW"BM32,2ØC2S8H4BH2HBG2F4BF
     F3BL4H2BHH2BL3F2BF2F3BD3H2BH3H2"
1040 GET(Ø,Ø)-(4Ø,38),PA,G
1050 DRAW"BM5Ø,3ØC3URURUR5S4UNR6
     US8R3URNE8UE7R2G2DG6LD2LD2FDGL
     GL2ULUEERE2L2DBM-3,-2D2G2L2U2"
1060 GET(5Ø,Ø)-(9Ø,38),SC,G
1070 DRAW"BM128,6L3GLGL2G3DF5R8E
     2UEU3H3LU":PAINT(124,16)
1080 DRAW"BM128,8C2L3GLGNL2DFRU
     ERER2U":PAINT(122,1Ø):DRAW"BM134,
     14L3GLNG3D2RNF2R4NG2EUH":PAINT
     (13Ø,18)
1090 GET(1ØØ,Ø)-(14Ø,38),ST,G
1100 E$="BR2BDNR3BD2R3BE3BR"
1102 A$(1)="NR2D4U2R4U2BF4U2NR2
     U2R4D4BR4U2NU2R5U2NL2BR8L4D2
     NR2D2R4BR3U4R4D2LDFRBE4"
1104 A$(2)="BD4R4U2L3U2R4BR8L4D4R4
     BR4U4BR8L4D2R3D2L3BR7R4U2L3U2R4
     BR3ND4R5D4NL2BR3U4R4D2LDFBR5
     R3U2L3U2R4BR2"
1110 A$(3)="BD4R4U2L3U2R4BR3R3
     ND4R2BR3NR5D4R5U4BR4ND4F3
     RFU4BF4NR4U2NR2U2R5BR3"
1120 I$="ND4BR6":Y$="C4F2ND2RE2
     BR3D4R4U4LBR6D4R3U4BR6"
1130 W$="D4RERERFRFU4BR4ND4BR5ND4
     F2RF2U4"
1140 S$="BD4R4U2L3U2R4BR3ND4R4
     D2NLD2BR4U4BR4D4R3EU2HBR9"
1150 D$="ND4R4D2NLD2BR7U4R2FD2
     GBR5U4R4D2LDFBR4U4R4D2LF2BR3NU4
     ERERFRFU4"
1200 RETURN
```

CUMULATIVE SUM

The cumulative sum program works by comparing the performance of the opponent over a period of time with sudden changes in response.

The graphics for each of these programs are the same as those used in the exponential smoothing programs. Check in the section for your machine to see exactly which lines are to be left alone.

RUN the program and enter a figure to regulate the computer's tactics.

Use Lines 52Ø to 61Ø from the previous program, and add this new section:

```
5 CLEAR 31999: GOSUB 5ØØ: BORDER Ø:
  PAPER Ø: INK 7: CLS
7 DIM A$(3,9): LET A$(1)="STONE": LET
  A$(2)="PAPER": LET A$(3)="SCISSORS"
10 DIM B(3): DIM U(3): DIM S(3): DIM
   H(3,2): DIM X(3): DIM Q(3): DIM P(3,3):
   DIM M(3)
15 LET MM=6Ø: DIM A(3,MM): DIM Z(MM)
20 FOR I=1 TO 3: LET H(I,1)=-COS
   ((I-2)*PI*2/3): LET H(I,2)=-SIN
   ((I-2)*PI*2/3): NEXT I
30 LET M(1)=1: LET M(2)=Ø: LET M(3)=Ø
40 INPUT "ENTER LIKELIHOOD RATIO FOR
   GAME (1 TO 9999), 1=NEW STRATEGY
   FOR EACH GAME, 9999=SAME
   STRATEGY.□";W
70 FOR I=1 TO 3: FOR J=1 TO 3: LET
   P(I,J)=SGN (I-J-3*INT((I-J+
   1.5)/3)): NEXT J: NEXT I: LET S=Ø
80 LET NN=Ø: LET U=1: LET S=Ø: LET
   U2=Ø: LET WW=Ø: LET U3=Ø
100 LET V=INT (RND*3)+1
110 PRINT INK 6; PAPER 2; "□□1=
    STONE, 2=PAPER, 3=SCISSORS□□"
120 INK 6: PLOT 8,167: DRAW 239,Ø: DRAW
    Ø,-159: DRAW -239,Ø: DRAW Ø,159
130 GOTO 4ØØ
200 LET Y=-1E3Ø: LET Z=-1E3Ø: FOR
    T=1 TO 3: LET I=U: IF T=1 THEN
    GOTO 23Ø
210 IF (U2=Ø AND T=2) OR (U3=Ø AND
    T=3) THEN GOTO 264
220 IF T=2 THEN LET I=ABS ((U=U2)-
    2*(U=VV)-3*((U<>U2) AND
    (U<>VV)))
```



```
230 IF T=3 THEN LET I=ABS ((U=U3)-
   2*(U=V3)-3*((U<>U3) AND
   (U<>V3)))
240 LET A(T,M(T))=I
250 FOR J=1 TO 3: LET B(J)=0: NEXT J:
   LET N=0: LET N2=M(1)
252 FOR M=M(T) TO 1 STEP -1: LET
   B(A(T,M))=B(A(T,M))+1: LET
   N=N+1: LET N2=N2-1
256 FOR J=1 TO 3: LET Q(J)=B(J)/N: LET
   X(J)=Q(J)+(Q(J)=0): NEXT J
258 LET Q=B(1)*LN (X(1))+B(2)*LN
   (X(2))+B(3)*LN (X(3)): IF NN>1 AND
   N2<>0 THEN LET Q=Q+Z(N2)
260 IF Q>Z THEN LET Z=Q: LET
   S(1)=Q(1): LET S(2)=Q(2): LET
   S(3)=Q(3): LET SS=T: LET NN=N2
262 NEXT M: IF Q>Y THEN LET Y=Q: LET
   TT=T: LET U(1)=Q(1): LET U(2)=Q(2):
   LET U(3)=Q(3)
264 NEXT T: LET T=TT: LET Z(M(1))=Y: LET
   Q(1)=U(1): LET Q(2)=U(2): LET
   Q(3)=U(3)
270 LET U2=U: LET VV=V: IF U2=VV
   THEN LET VV=U2+1-3*INT (U2/3)
272 IF U<>V THEN LET U3=U: LET
   V3=V: IF P(U3,V3)<0 THEN LET
   WW=U3: LET U3=V3: LET V3=WW
274 LET M(1)=M(1)+1: LET M(2)=M(2)+
   (U2>0): LET M(3)=M(3)+(U3>0)
280 IF Q>Z-LN (W) THEN GOTO 300
282 FOR T=1 TO 3: FOR M=NN+1 TO
   M(T): LET A(T,M-NN)=A(T,M): NEXT M
284 LET M(T)=(NN-M(T))*(-1*((M(T)>
   NN))): NEXT T
286 LET T=TT: LET Q(1)=S(1): LET
   Q(2)=S(2): LET Q(3)=S(3)
290 FOR M=1 TO M(1)-1: LET
   Z(M)=-1E30: NEXT M: LET Z(M(1))=Q
300 LET X=-1E30
310 IF T=1 THEN GOTO 350
311 IF T=2 THEN GOTO 320
312 IF T=3 THEN GOTO 340
320 LET WW=6-U2-VV: LET Q1=Q(1):
   LET Q2=Q(2): LET Q3=Q(3): LET Q(U2)
   =Q1: LET Q(VV)=Q2: LET Q(WW)=Q3
330 GOTO 350
340 LET WW=6-U3-V3: LET Q1=Q(1):
   LET Q2=Q(2): LET Q3=Q(3): LET Q(U3)
   =Q1: LET Q(V3)=Q2: LET Q(WW)=Q3
350 FOR G=1 TO 3: LET P=0: FOR I=1 TO
   3: LET P=P+P(G,I)*Q(I): NEXT I: IF
   P>X THEN LET X=P: LET V=G
360 NEXT G
400 INK 7: PRINT AT 2,4;"YOU SAID□"
405 FOR M=-3 TO 16: BEEP .01,M: NEXT M
410 LET K$=INKEY$: IF K$="" THEN
```

```
   GOTO 41Ø
412 IF K$<"1" OR K$>"3" THEN GOTO
   41Ø
415 LET U=VAL K$
42Ø PRINT AT 3,4;A$(U);AT 2,21; INK 5;"I
   SAID□";AT 3,21;A$(V)
43Ø POKE 23681,U-1: LET O=USR 32ØØØ:
   POKE 23681,127+V: LET O=USR 32ØØØ
44Ø LET S=S+P(U,V): PRINT AT
   16,8;"YOUR SCORE IS□";S;"□□"
45Ø INVERSE 1: IF V=U THEN PRINT AT
   18,1Ø;"□IT'S A DRAW□": GOTO 49Ø
46Ø IF (U=3 AND V=2) OR (U=2 AND
   V=1) OR (U=1 AND V=3) THEN PRINT
   AT 18,12;"□YOU WIN□": GOTO 49Ø
47Ø PRINT AT 18,11;"□□I WIN□□"
49Ø INVERSE Ø: FOR D=1 TO 2: PAUSE Ø:
   NEXT D: FOR N=2 TO 18: PRINT PAPER
   Ø; INK 7;AT N,2;"□□□□□□□□□□
   □□□□□□□□□□□□□□□□□□
   □□□": NEXT N: GOTO 2ØØ
5ØØ FOR N=32ØØØ TO 32284: READ A:
   POKE N,A: NEXT N
5Ø5 LET N=32Ø69: POKE 23728,N-256*
   INT (N/256): POKE 23729,INT (N/256)
51Ø RETURN
```

Use Lines 11Ø to 14Ø and Lines 6ØØ to 71Ø from the previous program.

```
4 FOR Z=12288 TO 12479:READ X:POKE Z,X:
   NEXT Z
5 SP=53248:POKE SP,8Ø:POKE SP+1,1ØØ:
   POKE SP+2,23Ø:POKE SP+3,1ØØ
6 POKE SP+23,255:POKE SP+29,255:POKE
   5328Ø,6:POKE 53281,3
7 A$(1)="STONE□□□":A$(2)="PAPER
   □□□":A$(3)="SCISSORS"
1Ø DIM B(3),X(3),U(3),S(3),Q(3),P(3,3),M(3),
   H(3,2)
15 MM=6Ø
2Ø DIM A(3,MM),Z(MM)
25 FORI=1TO3:H(I,1)=-COS((I-2)*π*
   2/3):H(I,2)=-SIN((I-2)*π*2/3):NEXTI
3Ø M(1)=1:M(2)=Ø:M(3)=Ø
4Ø PRINT"■■INPUT LIKELIHOOD RATIO
   TO BE USED":PRINT
   "PERMITTED RANGE 1 TO 9999"
5Ø PRINT"■■■■1□=□NEW
   STRATEGY FOR EVERY GAME":PRINT
   "9999□=□UNCHANGING STRATEGY"
6Ø PRINT"■SUGGESTED VALUE=2Ø■":
   INPUTW
7Ø FORI=1TO3:FORJ=1TO3:P(I,J)=SGN
   (I-J-3*INT((I-J+1.5)/3)):NEXTJ:
   NEXTI:S=Ø
8Ø U2=Ø:WW=Ø:U3=Ø
1ØØ V=INT(RND(Ø)*3)+1
2ØØ Y=-1E3Ø:Z=-1E3Ø:FORT=1TO3:
   I=U:IFT=1THEN23Ø
21Ø IF(U2=ØANDT=2)OR(U3=ØANDT=3)
   THEN264
22Ø IFT=2THENI=-(U=U2)-2*
   (U=VV)-3*((U<>U2)AND(U<>VV))
23Ø IFT=3THENI=-(U=U3)-2*
   (U=V3)-3*((U<>U3)AND(U<>V3))
24Ø A(T,M(T))=I
25Ø FOR J=1TO3:B(J)=Ø:NEXTJ:N=Ø:
   N2=M(1)
252 FORM=M(T)TO1STEP-1:B(A(T,M))=
   B(A(T,M))+1:N=N+1:N2=N2-1
256 FORJ=1TO3:Q(J)=B(J)/N:X(J)=
   Q(J)-(Q(J)=Ø):NEXTJ
258 Q=B(1)*LOG(X(1))+B(2)*LOG(X(2))+
   B(3)*LOG(X(3)):IFNN>ØTHEN
   Q=Q+Z(N2)
26Ø IFQ>ZTHENZ=Q:S(1)=Q(1):S(2)=
   Q(2):S(3)=Q(3):SS=T:NN=N2
262 NEXTM:IFQ>YTHENY=Q:TT=T:U(1)=
   Q(1):U(2)=Q(2):U(3)=Q(3)
264 NEXTT:T=TT:Z(M(1))=Y:Q(1)=U(1):
   Q(2)=U(2):Q(3)=U(3)
27Ø U2=U:VV=V:IFU2=VVTHENVV=U2+
   1-3*INT(U2/3)
272 IFU<>VTHENU3=U:V3=V:IFP(U3,V3)
   <ØTHENWW=U3:U3=V3:V3=WW
274 M(1)=M(1)+1:M(2)=M(2)-(U2>Ø):
   M(3)=M(3)-(U3>Ø)
28Ø IFQ>Z-LOG(W)THEN3ØØ
282 FORT=1TO3:FORM=NN+1TOM(T):
   A(T,M-NN)=A(T,M):NEXTM
284 M(T)=(NN-M(T))*(M(T)>NN):NEXTT
286 T=TT:Q(1)=S(1):Q(2)=S(2):Q(3)=S(3)
29Ø FORM=1TOM(1)-1:Z(M)=-1E3Ø:
   NEXTM:Z(M(1))=Q
3ØØ X=-1E3Ø
31Ø ONTGOTO35Ø,32Ø,34Ø
32Ø WW=6-U2-VV:Q1=Q(1):Q2=Q(2):Q3
   =Q(3):Q(U2)=Q1:Q(VV)=Q2:Q(WW)=Q3
33Ø GOTO35Ø
34Ø WW=6-U3-V3:Q1=Q(1):Q2=Q(2):
   Q3=Q(3):Q(U3)=Q1:Q(V3)=Q2:
   Q(WW)=Q3
35Ø FORG=1TO3:P=Ø:FORI=1TO3:
   P=P+P(G,I)*Q(I):NEXTI:IFP>XTHEN
   X=P:V=G
36Ø NEXTG
4ØØ PRINT "■■■■■■■■■
   YOU SAID■?□□□□□□□□□□■
   ■■■■■■■■■■■■■■■■■■";: GOSUB 7ØØ
41Ø GET A$:U=VAL(A$):IF U<1 OR U>3
   THEN 41Ø
42Ø PRINT A$(U);"■■■I SAID■";A$(V)
43Ø POKE 2Ø4Ø,191+U:POKE 2Ø41,191+V:
   POKE SP+21,3
44Ø S=S+P(U,V):PRINT "■■■■■
   ■■■■■"TAB(14)"■YOUR
   SCORE";S"■□□■■■■■"
45Ø PRINT "■";TAB(14);:IF U=V THEN
   PRINT "□IT'S A DRAW□":GOTO 49Ø
46Ø IF (U=3ANDV=2)OR(U=2ANDV=1)
   OR(U=1ANDV=3)THENPRINT "□□□
   YOU WIN□□□":GOTO 49Ø
47Ø PRINT "□□□□I WIN□□□□"
49Ø FOR Z=1 TO 999:NEXT Z:PRINT
   "■■"TAB(14);:FORZ=1 TO 13:PRINT
   "□";:NEXT Z
5ØØ PRINT:POKE198,Ø:GOTO 2ØØ
```

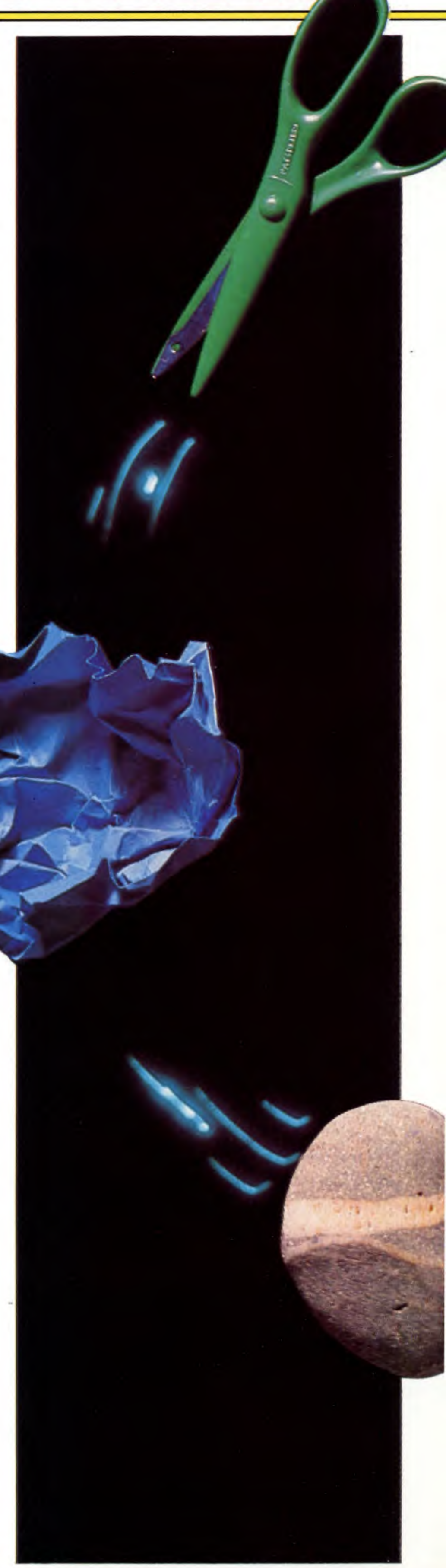
Delete up to 39Ø and RENUMBER from 51Ø.

```
1Ø CLS:F=Ø
2Ø DIMA$(3):A$(1)="STONE":A$(2)=
   "PAPER":A$(3)="SCISSORS"
3Ø MM=6Ø:DIMB(3),U(3),S(3),H(3,2),X(3),
   Q(3),P(3,3),M(3),A(3,MM),Z(MM)
4Ø FORT=1TO3:H(T,1)=-COS((T-2)*PI*
   2/3):H(T,2)=-SIN((T-2)*PI*2/3):NEXT
5Ø M(1)=1
6Ø INPUT"ENTER LIKELIHOOD RATIO OF
   GAME□□□□□□□□□□
   (1-9999), 1=NEW STRATEGY FOR
   EACH GAME, 9999=SAME STRATEGY.",
   W: IF W<1 OR W>9999 THEN 6Ø
7Ø FORT=1TO3:FORP=1TO3:P(T,P)=SGN
   (T-P-3*INT((T-P+1.5)/3)):NEXT:NEXT
8Ø S=Ø:U=1:NN=Ø:U2=Ø:WW=Ø:
   U3=Ø:V=RND(3)
9Ø MODE1
1ØØ PRINTTAB(5,1)"1=STONE□□□□
   2=PAPER□□□□3=SCISSORS"
11Ø IF F=ØTHENF=1:GOTO39Ø
12Ø Y=-1E3Ø:Z=-1E3Ø:FORT=1TO3:I=U
13Ø IF(U2=ØANDT=2)OR(U3=ØANDT=3)
   THEN23Ø
14Ø IFT=2THENI=-(U2=U)-2*
   (U=VV)-3*((U<>U2)AND(U<>VV))
15Ø IFT=3THENI=-(U3=U)-2*(U=
   V3)-3*((U<>U3)AND(U<>V3))
16Ø A(T,M(T))=I
17Ø FORJ=1TO3:B(J)=Ø:NEXT:N=Ø:
   N2=M(1)
18Ø FORM=M(T)TO1STEP-1:B(A(T,M))=
   B(A(T,M))+1:N=N+1:N2=N2-1
19Ø FORJ=1TO3:Q(J)=B(J)/N:X(J)=
   Q(J)-(Q(J)=Ø):NEXT
2ØØ Q=B(1)*LN(X(1))+B(2)*LN(X(2))+B(3)
   *LN(X(3)):IFNN>ØTHENQ=Q+Z(N2)
21Ø IFQ>Z□THENZ=Q:S(1)=Q(1):
   S(2)=Q(2):S(3)=Q(3):SS=T:NN=N2
22Ø NEXT:IFQ>Y□THENY=Q:TT=T:
   U(1)=Q(1):U(2)=Q(2):U(3)=Q(3)
23Ø NEXT:T=TT:Z(M(1))=Y:Q(1)=U(1):
   Q(2)=U(2):Q(3)=U(3)
24Ø U2=U:VV=V:IFU2=VV□THEN
   VV=U2+1-3*INT(U2/3)
25Ø IFU<>V□THENU3=U:V3=V:IFP(U3,
   V3)<ØTHENWW=U3:U3=V3:V3=WW
26Ø M(1)=M(1)+1:M(2)=M(2)-
   (U2>Ø):M(3)=M(3)-(U3>Ø)
27Ø IFQ>Z-LN(W) THEN32Ø
```

```
280 FORT=1TO3:FORM=NN+1TOM(T):
    A(T,M-NN)=A(T,M):NEXT
290 M(T)=(NN-M(T))*(M(T)>NN):NEXT
300 T=TT:Q(1)=S(1):Q(2)=S(2):Q(3)=S(3)
310 FORM=1TOM(1)-1:Z(M)=-1E3Ø:
    NEXT:Z(M(1))=Q
320 X=-1E3Ø
330 ONT□GOTO37Ø,34Ø,36Ø
340 WW=6-U-VV:Q1=Q(1):Q2=Q(2):
    Q3=Q(3):Q(U2)=Q1:Q(VV)=Q2:
    Q(WW)=Q3
35Ø GOTO37Ø
36Ø WW=6-U3-V3:Q1=Q(1):Q2=Q(2):
    Q3=Q(3):Q(U3)=Q1:Q(V3)=Q2:
    Q(WW)=Q3
37Ø FORG=1TO3:P=Ø:FORI=1TO3:
    P=P+P(G,I)*Q(I):NEXT:IF->X□
    THEN X=P:V=G:NEXT
39Ø PRINTTAB(4,4)"YOU SAID□";
4ØØ FORM=ØTO2ØSTEP2:SOUND1,-5,M*5,
    1:NEXT
41Ø K$=GET$
42Ø IFK$<"1" ORK$>"3" THEN 41Ø
43Ø U=VALK$
44Ø PRINTA$(U)"□□□I SAID□"A$(V)
45Ø VDU29,3ØØ;512;:PROCOBJ(U):VDU29,
    9ØØ;512;:PROCOBJ(V)
46Ø S=S+P(U,V):PRINTTAB(12,25)"YOUR
    SCORE IS□";S
47Ø IF V=U□THEN PRINTTAB(15,22)"IT'S
    A DRAW":GOTO5ØØ
48Ø IF(U=3ANDV=2)OR(U=2ANDV=1)
    OR(U=1ANDV=3)THENPRINTTAB(15,22)
    "YOU WIN":GOTO5ØØ
49Ø PRINTTAB(17,22)"I WIN"
5ØØ D=GET:GOTO9Ø
```

Delete to Line 435 and add these lines.

```
5 CLEAR1ØØØ:PMODE3,1:COLOR4,2:PCLS
6 GOSUB1ØØØ
7 C=8*ATN(1)/3
1Ø DIMB(3),U(3),S(3),H(3,2),X(3),Q(3),
   P(3,3),M(3)
15 MM=6Ø:DIMA(3,MM),Z(MM)
2Ø FORI=1TO3:H(I,1)=-COS((I-2)*C):
   H(I,2)=-SIN((I-2)*C):NEXT
3Ø M(1)=1
4Ø CLS:INPUT"ENTER LIKELIHOOD RATIO
   FOR GAME (1 TO 9999), 1=NEW
   STRATEGY FOR EACH GAME,
   9999=SAME STRATEGY.□";W
7Ø FORI=1TO3:FORJ=1TO3:P(I,J)=SGN
   (I-J-3*INT((I-J+1.5)/3)):NEXTJ,I
8Ø U=1
1ØØ V=RND(3)
11Ø COLOR4:PCLS:LINE(8,2Ø)-(247,171),
    PSET,B
12Ø DRAW"BM34,1ØS4C4":FORK=1TO3:
    DRAWN$(K)+E$+A$(K)+"BR4":NEXT
14Ø GOTO4ØØ
2ØØ Y=-1E3Ø:Z=-1E3Ø:FORT=1TO3:
    I=U:IFT=1 THEN23Ø
21Ø IF(U2=ØANDT=2)OR(U3=ØANDT=3)
    THEN264
22Ø IFT=2 THENI=-(U=U2)-2*(U=
    VV)-3*((U<>U2)AND(U<>VV))
23Ø IFT=3 THENI=-(U=U3)-2*
    (U=V3)-3*((U<>U3)AND(U<>V3))
24Ø A(T,M(T))=I
25Ø FORJ=1TO3:B(J)=Ø:NEXT:N=Ø:
    N2=M(1)
252 FORM=M(T)TO1 STEP-1:B(A(T,M))=
    B(A(T,M))+1:N=N+1:N2=N2-1
256 FORJ=1TO3:Q(J)=B(J)/N:X(J)=
    Q(J)-(Q(J)=Ø):NEXT
258 Q=B(1)*LOG(X(1))+B(2)*LOG(X(2))+
    B(3)*LOG(X(3)):IFNN>Ø THEN
    Q=Q+Z(N2)
26Ø IF Q>Z THENZ=Q:S(1)=Q(1):S(2)=
    Q(2):S(3)=Q(3):SS=T:NN=N2
262 NEXTM:IFQ>Y□THENY=Q:TT=T:
    U(1)=Q(1):U(2)=Q(2):U(3)=Q(3)
264 NEXTT:T=TT:Z(M(1))=Y:Q(1)=U(1):
    Q(2)=U(2):Q(3)=U(3)
27Ø U2=U:VV=V:IFU2>VV□THEN
    VV=U2+1-3*INT(U2/3)
272 IFU<>V□THENU3=U:
    V3=V:IFP(U3,V3)<Ø THEN
    WW=U3:U3=V3:V3=WW
274 M(1)=M(1)+1:M(2)=M(2)-(U2
    >Ø):M(3)=M(3)-(U3>Ø)
28Ø IFQ>Z-LOG(W) THEN3ØØ
282 FORT=1TO3:FORM=NN+1
    TOM(T):A(T,M-NN)=A(T,M):NEXT
284 M(T)=(NN-MT)*(M(T)>NN):NEXT
286 T=TT:Q(1)=S(1):Q(2)=S(2):Q(3)=
    S(3)
29Ø FORM=1TOM(1)-1:Z(M)=-1E3Ø:
    NEXT:Z(M(1))=Q
3ØØ X=-1E3Ø
31Ø ON T GOTO35Ø,32Ø,34Ø
32Ø WW=6-U2-VV:Q1=Q(1):Q2=Q(2):
    Q3=Q(3):Q(U2)=Q1:Q(VV)=Q2:
    Q(WW)=Q3
33Ø GOTO35Ø
34Ø WW=6-U3-V3:Q1=Q(1):Q2=Q(2):
    Q3=Q(3):Q(U3)=Q1:Q(V3)=Q2:
    Q(WW)=Q3
35Ø FORG=1TO3:P=Ø:FORI=1TO3:
    P=P+P(GI)*Q(I):NEXT:IFP>X□
    THEN X=P:V=G
36Ø NEXT
4ØØ SCREEN1,Ø:DRAW"BM12,3Ø"
    +Y$+S$
41Ø A$=INKEY$:IFA$<"1" ORA$>"3"
    THEN41Ø
42Ø U=VAL(A$):DRAWA$(U)+
    "BM142,3Ø"+I$+S$+A$(V)
43Ø X=44:Y=9Ø:ON U GOSUB6ØØ,61Ø,62Ø
435 X=17Ø:ON V GOSUB 7ØØ,71Ø,72Ø
```



BUILDING UP FORTH

With effective programs that run more quickly than the equivalent programs in BASIC, FORTH is fast gaining popularity as a general-purpose language

This, the second part of the series on FORTH, examines the contents of the FORTH dictionary and how these words are employed.

FORTH words can be used to do a simple operation on data held on the stack—or on the contents of specific memory locations. They can also initiate a sequence of separate operations—in effect working as a program. Variables, constants, arrays and buffers may be used more or less as in any other type of language.

CONSTANTS AND VARIABLES

The stack is fine for temporary storage of numbers but different measures have to be taken to deal with anything more permanent. For this, constants and variables are used. Both are given dictionary entries as separate words, but unlike in BASIC, they are quite different data types (in BASIC a constant is a fixed-value variable, nothing more).

FORTH already holds several defined constants which are simply keyed in as required, or you can easily define your own. Variables are used extensively, and in very much the same way as in BASIC. Several variables are defined in the dictionary and these will be looked at later on.

But a significant use of variables is for values which are used inside definitions which have already been compiled and may need to be adjusted during use.

CONSTANT and VARIABLE are both *defining words*, which simplifies the process of assigning new values to chosen constants or variables. A constant may be defined using the following command format:

<value> CONSTANT <constant name>

Thus if you type:

365 CONSTANT YEAR <RETURN>

and press RETURN or ENTER, the screen displays:

365 CONSTANT YEAR OK

The computer assigns the value 365 to the constant YEAR, so 365 is produced—and pushed onto the stack—each time YEAR is executed.

Variables are treated similarly and, as in BASIC, they may be assigned values at the start or during a program. The variable may be defined using the command format:

<value> VARIABLE <variable name>

The dictionary definition of a constant or variable consists of a header and a two-byte body which stores a 16-bit value. During execution a constant fetches the value stored in these two bytes and places it on the stack. A variable, on the other hand, puts the location address of the body value on the stack as soon as that variable name is executed. That's the difference between the two.

In addition there are a number of user variables but the contents of these are not



- CONSTANTS AND VARIABLES
- USER VARIABLES
- FETCHING AND STORING
- MORE FIGURES
- NEW DEFINITIONS
- COLON DEFINITION
- SCREENS
- MORE ON OUTPUT
- DOT QUOTE
- INPUT

stored immediately following the header in memory. Instead, they are placed near the top of memory in a table called the user area. When a user variable is executed, its address is left in its relevant slot in the user area but otherwise it is functionally similar.

FETCHING AND STORING

The fetching and storing of information from memory locations, such as for variables, is undertaken by versions of two words: @ (*fetch*) and ! (*store*) which are directly comparable to PEEK/POKE in BASIC.

@, in fact, reads the 16-bit (two byte) contents of the memory location pointed to by the address on the stack. As executing a variable leaves its contents on the stack, @ is a convenient way to read the contents of a variable.

One of the user variables is the word BASE which is used for keeping tabs on the current number base being used—but which defaults to 1Ø.

So, for example, if you wanted to check on the current value, all that's needed is:

```
BASE @ . <RETURN>
```

and this would display:

```
BASE @ . 1Ø OK
```

The word ! requires two values on the stack in order to write a value to a 16-bit memory location. The topmost value is the storage address, the second figure the value that is to be stored—both are removed from the stack when ! has been executed.

Single byte values—rather more usable on 8-bit machines—are fetched using the word C@ and stored with C!.

Suppose you wanted to turn the screen black on a Commodore 64. In BASIC this requires the value Ø to be POKEd into location 5328Ø. The equivalent in FORTH is achieved by keying in:

```
Ø 5328Ø C!
```

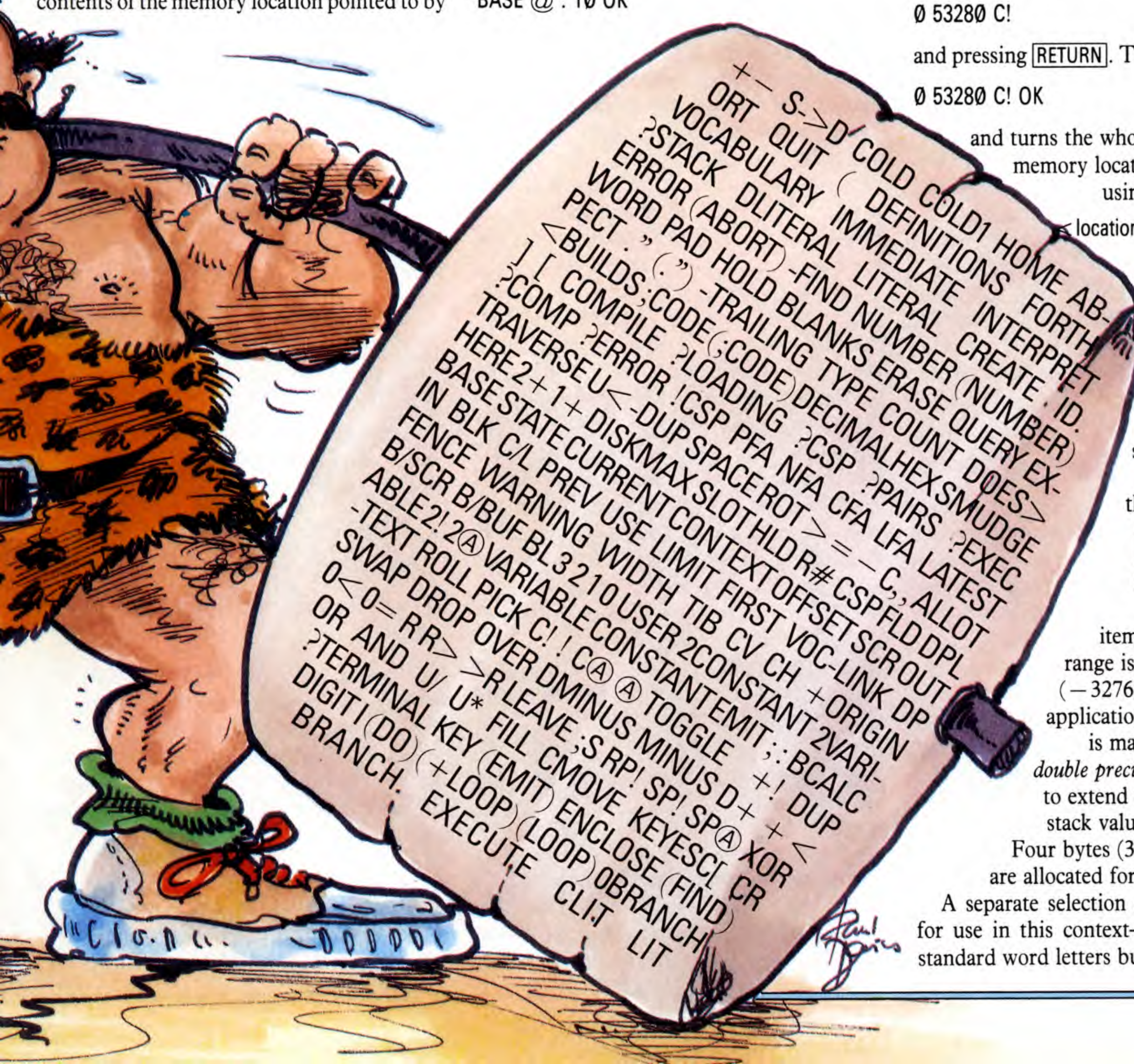
and pressing RETURN. This gives the display:

```
Ø 5328Ø C! OK
```

and turns the whole screen black. The memory location can be inspected using C@ in the format:

```
<location> C@ <RETURN>
```

Commodore users familiar with the screen colour change POKE referred to here may be interested to know that only a single POKE to 5328Ø is required, rather than to 53281 as well.



MORE FIGURES

One of the problems of using 16-bit data items is that the number range is rather too restricted (−32768 to 32767) for some applications. Special provision is made to get round this: *double precision integers* are used to extend the range of possible stack values to ±2147483648. Four bytes (32 bits) of stack space are allocated for these numbers.

A separate selection of words is reserved for use in this context—they're usually the standard word letters but prefixed by D or 2.

Most of the stack manipulation commands for double precision work begin with 2, whereas arithmetic operations begin with D.

Thus a double number can be removed from the stack using the word D. (*d-dot*) and you've also got words like 2CONSTANT and 2VARIABLE which can be used to handle really big number values. But some words work with single or double numbers.

NEW DEFINITIONS

The real power of FORTH is the way you can create new instructions, literally on the back of existing word definitions. Very complex—but customized—data storage commands and manipulations are thus made possible.

The nature of word definitions suggests a kind of tree-like structure where the most 'recent' word can be argued to branch from several others. In fact, FORTH adopts a linear structure where the latest word definition is placed on top of the dictionary in the CURRENT vocabulary. Pointers are provided in the header of the latest word so that it links to the previously defined word upon which it might be based. Because of this linear structure, FORTH is sometimes referred to as a threaded interpretive language.

To recap, the usual way to add new commands to a 'personalised' FORTH system is to create a selection of *colon definitions*, essentially a list of pointers to the existing, previously defined, words available.

A colon definition starts with a colon, is followed by a space and the definition's chosen name, followed in the *body* by a space and then the name(s) of previously defined FORTH words which have been used to construct that definition. The entry is terminated by a further space and then a semi-colon:

: <new definition name> <body —old names>;

Spaces are very important separators and, as a general rule, must always be used when entering separate data items, operators and words anywhere in the input stream.

FINDING A PLACE

As soon as the colon definition is entered it is added to the top of the CURRENT vocabulary in the dictionary. The FORTH system stops interpreting or executing commands when it comes across a colon in the input stream. The system then enters compile mode. A *header entry* is created for the new definition, its name is assigned to that header, and all the values and 'old' words and operators used for the definition are said to be 'compiled into the definition'.

If a pointer is added into the body, upon execution of the new word, the computer is directed to another definition to execute, or to a routine that will place the required numeric value onto the stack.

The new word, once defined, may itself be nested in the definitions of further words, at any depth (in any part of the body), and as often as required.

Remember, you can use VLIST to inspect what's in the dictionary, including the latest words you've added to the CURRENT vocabulary which will be at the very start of the list. A typical list of words is shown alongside, and it's interesting to see how some of the latest definitions (those nearer the start) resemble the earlier definitions on which they were clearly based. Any of the words may be removed from the CURRENT vocabulary by using the command:

FORGET <word you want removed>

Some caution is required in the use of this word, however, as definitions operate on a pointer system back to previous words, FORGET not only wipes out the chosen word, but also any others based on it!

Almost all the dictionary words may be used in colon definitions, but there are some exceptions which will throw up a quite obvious error message. Incidentally, if the input stream or part of it is rejected, FORTH will usually display the offending character sequence (word) along with an error message symbol, normally a question mark.

As many words as you like may be used to make up the body of a new definition. Although it's considered good practice to keep word definitions short and therefore very transportable and usable, it's conceivable that the input line maximum may be reached. No problem: FORTH is free format which means you can enter separate lines. Simply press RETURN or ENTER after each line and start the next line with]. Some systems stay in compile mode until the semicolon is reached and a square bracket is not needed.

When compilation is complete, you lose the *source code* used to construct the body of the new definition. This may be a problem if you ever wish to examine the word sequence and algorithms employed, perhaps to eliminate a bug.

SCREENS

The concept of *screens*, the device for recording essential background data for a definition, has already been mentioned. Everything to do with a new definition—old definitions, constant and variable declarations, arrays—can be logged in a screen. And the screen can then be saved, and reloaded when required for editing, or as an input stream.

The word LOAD reads in a source screen and treats it as if it was a normal input line from the keyboard. It's used in this form:

n LOAD

where n is the designated screen number. Each one can provide up to 1024 bytes of data. A *screen editor* (if not resident, this is normally available in source code format—on screens!) enables you to revise the source code definitions for new words within the screen area, and these may be resaved to disk. This is much easier than using a simple line editor to amend screen data.

The 1024 byte screen input stream is not a handicap: one screen may load another using the command word — — > (*next screen*) and an entire program, with source code on many screens, may be loaded with a single one-word command. Loading terminates when the word ;s (*semi-s*) is encountered.

When a screen is used for the input stream, anything that isn't defined in the vocabularies is executed immediately—so screenfuls of data may be read or written directly to or from memory. This permits screens to be used for, amongst other things, storing data arrays, tables, overlays and machine code routines.

MORE ON OUTPUT

Communication with a FORTH program requires certain input and output commands, some of which have been seen already. Let's look at output first.

The basic output word used in FORTH is EMIT which is used in the form:

<ASCII character code> EMIT

This transfers the ASCII character to the stack and is similar to PRINTCHR$ in BASIC. Thus if you enter:

72 EMIT 69 EMIT 76 EMIT 76 EMIT 79 EMIT

the screen displays:

72 EMIT 69 EMIT 76 EMIT 76 EMIT 79 EMIT HELLOOK

The process may be a bit clumsy in this instance, perhaps—particularly in terms of the display. In fact, there is an easier way to print out ASCII messages, and this is done using ." (*dot quote*) in the command format—note that there must be a space after the first quote marks:

."□ <text string>"

So entering:

."□THIS IS AN EXAMPLE"



produces the display:

." THIS IS AN EXAMPLE" THIS IS AN EXAMPLE OK

In passing, it's worth mentioning that dot-quote may be used within a colon definition to print a screen message when the definition is executed.

Also, display formatting of string or numeric output is possible and several special words exist for this purpose.

INPUT

Now what of input? There is so much scope that it's impossible to be anything but brief in this general introduction of FORTH. Single characters, strings and numerals can, of course, be entered but involve some 'behind the scenes' thinking. Command words start things off.

A string of characters may be entered using the EXPECT command. This word works in very much the same way as INPUT in BASIC. But before the word EXPECT is executed, the program must indicate just where in memory the string of characters are to be placed, as well as the maximum number of characters to expect, although this is only a problem on entries longer than a single screen or logical line.

During execution, EXPECT waits until the specified maximum number of characters has been reached—or until [RETURN] or [ENTER] is pressed and then null bytes are used to fill up the input string storage space allocation (which is typically a default of 40 characters).

Although EXPECT has its uses, there's a better word—WORD. This reads the next text character from the input stream, and continues until a delimiter is found. This is specified immediately prior to the use of WORD and is simply the character that the text string may start and end with—this could be a space, quote character, or anything. The string is packed in the same way as for EXPECT and it is stored at the address given by HERE, a word which returns the current top of the dictionary (the first free byte at the end of the dictionary).

Numeric input is simple enough before a FORTH word is executed, but a special routine has to be defined to get—and possibly convert—user input. In BASIC this is done automatically using a line such as:

INPUT"ENTER NUMBER";N

A special word is used: NUMBER. This converts an ASCII string into a (signed) double number which it leaves on the stack. The conversion is to the current number base (set using BASE) and stops as soon as a null byte or space is reached in the string. To get round this, EXPECT is used to input the string. If a decimal point is encouraged, the NUMBER routine does some 'housekeeping' and stores its position in one of the system's user variables (DPL) where it will remain ignored unless required.

Ultimately, you've got to define a special word which embraces NUMBER and EXPECT. A typical definition for numeric input could take the form:

: INPUT DECIMAL (sets decimal base)
HERE 1 + 10 EXPECT (set to read up to ten characters)
10 HERE C! (place the byte count at start)
HERE NUMBER ; (converts the string to a number)

A single precision integer can be input using the same routine by placing the command word DROP (and spaces!) just before the semi-colon.

WORD may also be used to input numeric values, so it is possible to read in and convert numeric strings belonging to a table.

In the definition above for INPUT the input stream could contain *comment lines* similar to those alongside. These are comparable to REM statements in BASIC and are obviously extremely useful for annotating listings of display screens. Round brackets must be used and a space must follow the lefthand one.

The next article takes a look at FORTH control structures—which involves logical operations and decision processes.

A PROGRAM CROSS-REFERENCER

This handy machine code utility can find any string or keyword in a BASIC program in seconds, saving you hours of time and effort. It will even replace strings for you

The cross-referencer program that accompanies this article is known as a utility program. This means that it doesn't do anything on its own, but instead, is used as a tool to help you develop other programs. For this reason the cross-referencer is written in machine code so that it can be loaded into the computer at the same time as a BASIC program. The machine code then remains in the computer's memory waiting to be called up at any time you need its help.

The cross-referencer acts as a retriever, to save you from having to hunt for a particular point in your program. If you tell it what you are looking for—a string, a keyword, a variable name or whatever—it will search through your BASIC program and print out all lines in which it occurs. As well as this it has an extra 'replace' facility which allows you to replace one string with another—throughout your program.

The utility is most useful when you are developing or writing a BASIC program. For example, if you decide you need to change some of the PRINT or INPUT statements, the program will list all the lines for you. Or if you realize that the name of one of your variables is going to clash with another routine, you can choose the replace option and change the name to something else—without you having to edit a single line.

The cross-referencer is also useful to have in your computer when you are trying to adapt or understand someone else's program. For instance, it can be a revealing exercise to print out all lines in which the word GOTO or GOSUB appeared, as you can quickly trace through the sequence of events.

It will also help you track down bugs in your program. For example, you might discover that a certain variable is taking on an incorrect value somewhere in your program. The usual process is to search through the program line by line to find every place where the variable was used—and it is very easy to miss a few. The cross-referencer, though, will do the job for you accurately, in a matter of seconds.

As usual, the versions for each computer are slightly different and so are explained separately below.

S

The Spectrum cross-referencer is part of a larger toolkit program, the remainder of which will be given in part two. However, the cross-referencer is quite independent and can be used on its own if you like. The program consists of a short BASIC loader program followed by the machine code in DATA statements. Type in the BASIC program, SAVE it and RUN it, and, assuming all is well, the program will SAVE the resulting machine code as well.

When you come to use the program it is this machine code version that you'll need. To load it, type:

```
CLEAR 64559
LOAD "CREF" CODE
```

You can load it in before or after the BASIC program you want to work on—the procedure is the same in both cases. Once loaded, you can call up the various options using the RANDOMIZE USR calls as detailed in the following paragraphs.

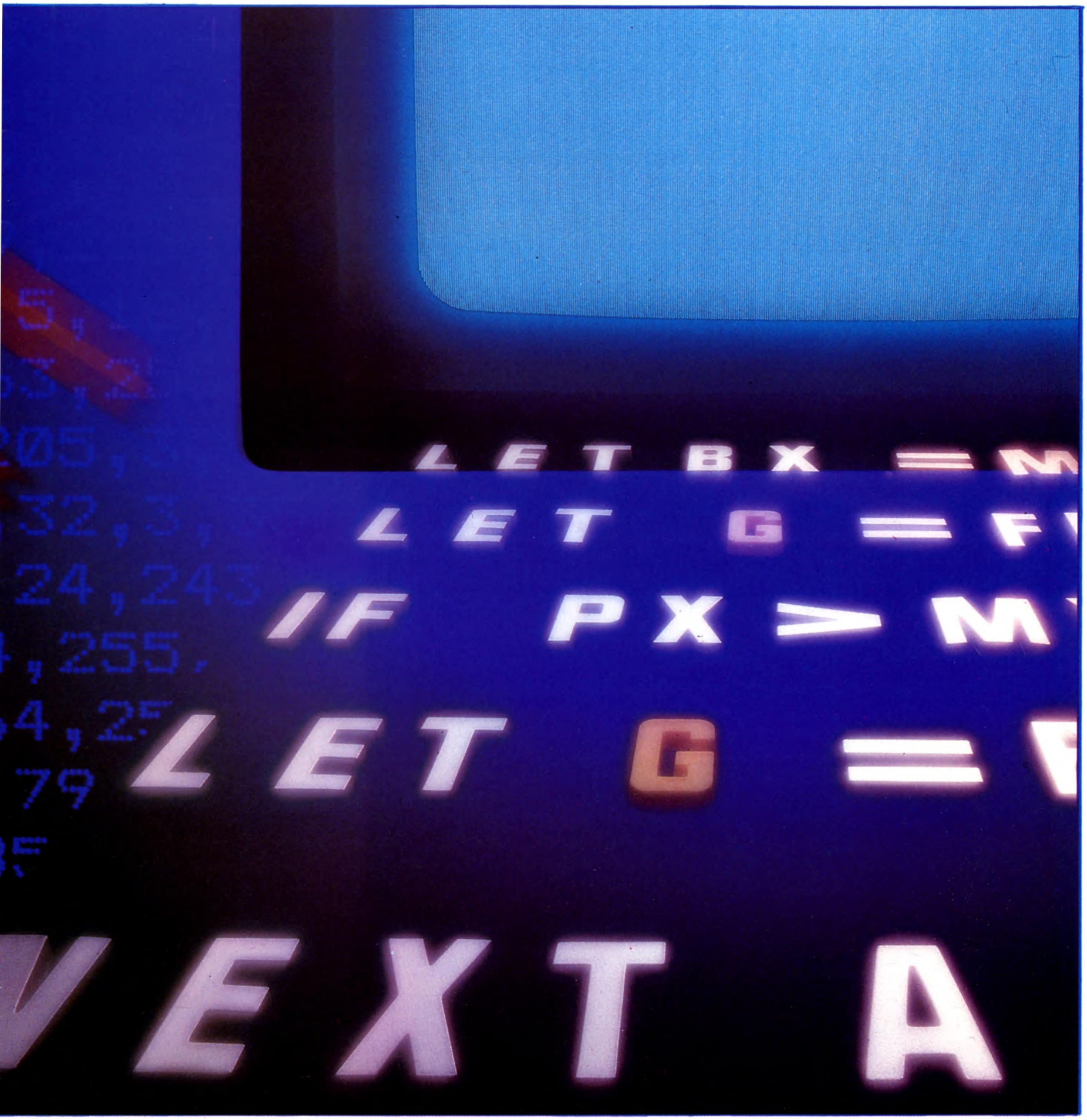
To search for a string, type RANDOMIZE USR 64634. You'll see the prompt 'enter target string', and when you've typed in the string you're looking for and pressed ENTER, the program will print out all lines in which it occurs.

To search for a BASIC keyword, type RANDOMIZE USR 64911. Again you'll be prompted to enter the target string—in this case the keyword you're looking for—and the program will print out all lines containing this keyword.

To replace the name of a variable, type RANDOMIZE USR 64796. This time you have to enter both the target string and the replacement string. These can have a maximum of 20 characters and the replacement string can be either longer or shorter than the original one. The replacement that you have specified is made throughout the program automatically, but this time the lines are not printed up on the screen.

The last option allows you to see all lines in which a function was defined (using DEFFN) or a function was called. To use this option type RANDOMIZE USR 64713.




LET BX =M
LET G =F
IF PX > M
LET G =
NEXT A

The Commodore cross-referencer is part of a suite of toolkit programs. Two more to follow are a general toolkit offering many extra commands including Renumber and Auto, and a disk editor program. The program consists of a short BASIC loader program and the machine code in DATA statements. Type in the program and then SAVE it before you RUN it in case it crashes. Then RUN it, and, assuming all is well, SAVE the resulting machine code using:

```
POKE 43,0 :POKE 44,192 :POKE 45,47 :POKE
    46,195 :CLR
SAVE "CREF",1,1
```

or, if you prefer, you can use the machine code monitor on page 280.

When you come to use the program it is this machine code version that you'll need. To LOAD it type:

```
LOAD "CREF",1,1
```

You should load it in at the start of a programming session, before you load in a BASIC program. Call it up at any time using SYS 49152, the following options are then available. Remember if you are using a disk drive you should change the first figure 1 in the LOAD and SAVE commands to an 8.

To search for a string, type L (for List) followed by the characters you're searching for—up to a maximum of 76 characters—then press RETURN. The program will print out all lines in which the string occurred.

To replace a string, type R followed by the original string then a slash (/) then the replacement string—for example R ab/zz. A maximum of four characters can be replaced and the replacement string can be shorter but not longer than the original.

Typing K followed by the name of a keyword will list all lines containing the keyword.

Typing F followed by the name of the function will list all lines in which the user-defined function is either defined or called.

Finally, type E to exit the program and return to BASIC.

The Acorn program is given in assembly language rather than machine code. This means it is quite long but has the advantage of being easier to read. Type in and SAVE the whole program. Once SAVEd you should RUN the program to generate a machine code version and then SAVE this as well following the instructions given in the program. It is the machine code rather than the assembly language program that you need to LOAD into your computer each time you wish to use the cross-referencer.

The machine code is stored at location &5000 which is just at the top of BASIC under the screen memory. If you want to alter this location, change the value of base% in Line 10 of the program.

To use the program type *LOAD"CREF" (or whatever name you've called the machine code version) at the start of a programming session then LOAD in your BASIC program normally. The cross-referencer is called with CALL &5000. After this, you'll see a menu offering eight options, all of which are very straightforward to use.

If the output of the cross-referencer moves by too quickly for you then the computer should be put in paged mode. This can be done by typing CTRL N in response to the double arrow prompt. Similarly, paged mode can be turned off with CTRL O.

MENU OPTIONS

The menu offers the following options:

S: Search for a string. This allows you to search through a program held in the computer's memory between the current values of PAGE and TOP, for a given string (not including keywords). Whenever the cross-referencer asks you to enter a string a standard routine is used. This is arranged to only accept characters with ASCII codes between 32 and 128; the maximum string length is 50 characters. In entering the string, DELETE, CTRL U, SHIFT and the cursor keys can be used. Pressing ESCAPE will return to the main menu. Any line in which it occurs will be printed out.

R: Replace a string with another. This asks you for two strings. A search of the program is then done and every time a copy of the first string is found, the line in which it occurs will be printed out. You will then be asked if you wish to have the string replaced. The line is then listed again. If the search string occurs more than once in a line you will be asked about replacement for each occurrence.

K: Search for a keyword. When you INPUT a keyword, all lines in which the keyword occurs will be listed.

L: Search for a line number. In BBC BASIC, line numbers which appear after GOTO, GOSUB and ON are tokenized (like keywords) so a separate option is provided to look for them.

C: Change a line number. This is just like the R option in that it allows the replacement of line numbers. You will be asked to INPUT two line numbers and will be given the chance to change each occurrence.

F: List Functions. This prints out all lines containing a function definition or call.

P: List Procedures. This prints out all lines containing a procedure definition or call.

X: Exit the program. This returns you to the BASIC program.

The Dragon and Tandy program consists of a short loader program followed by the machine code in DATA statements. Type the program in very carefully. The computer will print the message 'check sum error' if you make a mistake in the DATA lines.

CSAVE the BASIC program once it runs without an error message and then save the machine code using:

```
CSAVEM"CREF",28672,29500,28672
```

It is the machine code version that you'll be using. The top of BASIC is lowered by 4K in order to store the program so it cannot be used with very long programs that extend above 6FFF.

When you want to use the program load it back by typing:

```
CLEAR 200,&H6FFF
CLOADM "CREF"
```

It doesn't matter whether you load in the cross-referencer before or after you load in the BASIC program. The procedure is exactly the same. To call the program type EXEC &H7000 in direct mode and you'll see a menu offering the following options: (S)trings, (R)eplace, (K)eywords, (F)unctions. Press the initial letter of the option you want.

Press S to search for any string in the program. You will see the prompt FIND?. When you have entered what you are looking for, the program will search through your BASIC program and list all lines where a match is found. If no matches are found NOT FOUND is printed and the computer returns to BASIC.

(R)eplace asks you which string you want to replace and the new string to be inserted. The new string must contain the same number of characters. The replacement is automatic and the new lines are relisted.

Pressing K or F allows you to search for BASIC words. When using these options you must keep in mind the distinction between keywords such as PRINT and GOTO and functions such as VARPTR and SQR. If you are unsure which is which, the simplest solution is to try both options.

If you have a Microdrive, change the word 'cassette' in Line 30 to 'cartridge' and change

Line 87 to 87 SAVE"'m"; 1; "CREF" CODE 64560, 832.

```
10 CLEAR 64559: BORDER 0: INK 7: PAPER
   0: CLS
20 PRINT INVERSE 1;AT 0,7;"□CROSS
   REFERENCER. "
30 PRINT ''''"□Poking in machine code.
   Prepare a casette for saving."
40 LET L=90: RESTORE L: FOR N=64560
   TO 65343 STEP 16
50 LET T=0: FOR D=0 TO 15: READ A:
   POKE N+D,A: LET T=T+A: NEXT D
60 READ A: IF A<>T THEN PRINT FLASH
   1;"□CHECK-SUM ERROR IN
   LINE ";L;"!": PRINT "EXPECTED
   VALUE ";A;"ACTUAL VALUE□";T: STOP
70 LET L=L+10: NEXT N
80 PRINT AT 18,0;"ANY KEY TO START SAVE
   PROCEDURE"
85 LET A$=INKEY$: IF A$="" THEN GOTO
   85
87 SAVE "CREF"CODE 64560,832
88 STOP
90 DATA 203,39,95,22,0,221,33,233,253,221,
   25,221,110,0,221,102,1999
100 DATA 1,205,69,252,201,62,254,229,205,
   1,22,225,126,35,254,255,2396
110 DATA 40,4,215,195,76,252,201,207,9,
   209,225,205,229,25,201,205,2498
120 DATA 142,2,14,0,32,249,205,30,3,48,244,
   21,95,205,51,3,1344
130 DATA 254,0,201,46,25,118,45,32,252,
   201,62,13,205,48,252,33,1787
140 DATA 63,255,17,85,255,205,3,253,42,83,
   92,34,59,255,126,254,2081
150 DATA 64,208,17,4,0,25,126,254,13,32,3,
   35,24,237,205,179,1426
160 DATA 252,56,3,35,24,240,229,42,59,255,
   205,85,24,62,13,215,1799
170 DATA 225,24,240,229,17,85,255,58,63,
   255,71,26,19,190,35,32,1824
180 DATA 5,16,248,225,55,201,225,191,201,
   62,254,205,1,22,42,83,2036
190 DATA 92,34,59,255,126,254,64,208,17,4,
   0,25,126,254,13,32,1563
200 DATA 3,35,24,237,254,168,40,7,254,206,
   40,3,35,24,237,229,1796
210 DATA 42,59,255,205,85,24,62,13,215,225,
   62,13,1,0,0,237,1498
220 DATA 177,24,206,175,119,213,229,205,
   115,252,205,95,252,245,215,241,2968
230 DATA 225,209,254,13,40,5,18,19,52,24,
   234,201,62,13,205,48,1622
240 DATA 252,17,85,255,33,63,255,205,3,253,
   62,18,205,48,252,33,2039
250 DATA 64,255,17,65,255,205,3,253,42,83,
   92,126,254,64,208,17,2003
260 DATA 4,0,25,126,254,13,32,3,35,24,240,
   254,34,32,9,35,1120
270 DATA 1,0,0,237,177,126,24,243,205,179,
   252,56,3,35,24,227,1789
280 DATA 58,63,255,79,58,64,255,145,56,25,
   40,8,79,6,0,229,1420
290 DATA 205,85,22,225,58,64,255,17,65,255,
   79,6,0,235,237,176,1984
300 DATA 235,24,192,237,68,79,6,0,229,205,
   232,25,225,24,229,62,2072
310 DATA 13,205,48,252,17,85,255,33,63,255,
   205,3,253,33,150,0,1870
320 DATA 27,235,203,254,235,1,165,90,197,
   205,179,252,56,13,35,203,2350
330 DATA 126,35,40,251,193,12,16,240,195,
   87,252,193,42,83,92,34,1891
340 DATA 59,255,126,254,64,208,35,35,35,35,
   126,35,254,13,40,239,1813
350 DATA 185,32,247,197,62,13,1,0,0,237,
   177,229,42,59,255,205,1941
360 DATA 85,24,62,13,215,225,193,24,214,15,
   254,70,254,79,254,86,2067
370 DATA 254,98,254,110,254,27,254,35,254,
   48,254,64,254,124,254,143,2681
380 DATA 254,160,254,184,254,206,254,229,
   254,236,254,0,255,10,255,66,3125
390 DATA 89,84,69,83,32,70,82,69,69,61,255,
   80,82,79,71,82,1357
400 DATA 65,77,255,78,85,77,66,69,82,32,65,
   82,82,65,89,255,1524
410 DATA 67,72,65,82,65,67,84,69,82,32,65,
   82,82,65,89,255,1323
420 DATA 66,89,84,69,83,255,80,82,79,71,82,
   65,77,61,255,32,1530
430 DATA 66,89,84,69,83,255,70,73,76,69,32,
   84,89,80,69,61,1349
440 DATA 32,255,70,73,76,69,32,78,65,77,69,
   61,32,255,70,73,1387
450 DATA 76,69,32,76,69,78,71,84,72,61,32,
   255,69,78,84,69,1275
460 DATA 82,32,83,84,65,82,84,32,76,73,78,
   69,32,58,255,69,1254
470 DATA 78,84,69,82,32,69,78,68,32,76,73,
   78,69,32,58,255,1233
480 DATA 69,78,84,69,82,32,76,73,78,69,32,
   73,78,67,82,69,1111
490 DATA 77,69,78,84,83,32,58,255,69,78,84,
   69,82,32,84,65,1299
500 DATA 82,71,69,84,32,83,84,82,73,78,71,
   58,32,255,69,78,1301
510 DATA 84,69,82,32,68,69,67,73,77,65,76,
   32,78,85,77,66,1100
520 DATA 69,82,32,58,255,72,69,88,32,61,32,
   255,69,78,84,69,1405
530 DATA 82,32,72,69,88,39,32,78,85,77,66,
   69,82,58,32,255,1216
540 DATA 68,69,67,73,77,65,76,61,32,255,69,
   78,84,69,82,32,1257
550 DATA 78,69,87,32,83,84,82,73,78,71,58,
   255,0,0,0,0,1050
560 DATA 9,23,220,10,254,21,206,11,254,80,
   3,23,220,10,215,24,1583
570 DATA 177,1,0,10,0,100,0,232,3,16,39,48,
   48,49,48,0,771
```

```
0 DATA 234,169,62,32,12,225,169,0,133,247,
  133,79,133,100,169,#1897
10 DATA 8,133,101,32,18,225,201,76,240,50,
   201,82,240,107,201,#1915
20 DATA 70,240,31,201,69,240,26,201,75,
   240,99,169,69,32,12,#1774
30 DATA 225,169,82,32,12,225,32,12,225,169,
   13,32,12,225,76,#1541
40 DATA 0,192,234,96,234,162,0,169,165,
   149,85,232,76,78,192,#2064
50 DATA 234,162,0,234,32,38,194,160,0,234,
   177,100,201,0,208,#1974
60 DATA 7,32,158,192,201,0,240,158,234,
   162,0,213,85,208,24,#1914
70 DATA 165,101,133,102,132,103,234,228,
   96,240,86,232,32,28,194,#2106
80 DATA 177,100,213,85,208,3,76,111,192,
   234,32,28,194,76,84,#1813
90 DATA 192,234,76,60,194,162,0,32,18,225,
   201,13,240,6,149,#1802
100 DATA 85,232,76,142,192,76,202,194,234,
   32,28,194,177,100,201,#2165
110 DATA 0,208,28,32,28,194,177,100,201,0,
   240,18,32,28,194,#1480
120 DATA 165,101,133,95,152,133,94,32,28,
   194,32,28,194,177,100,#1658
130 DATA 96,32,28,194,76,177,192,234,165,
   247,201,1,208,3,32,#1886
140 DATA 127,194,165,94,168,165,95,133,
   101,169,0,133,100,177,100,#1921
150 DATA 133,252,32,28,194,177,100,133,
   253,72,138,72,152,72,32,#1840
160 DATA 181,193,104,168,104,170,104,32,
   28,194,234,177,100,201,34,#2024
170 DATA 240,38,201,0,240,25,201,255,240,
   18,234,201,127,176,49,#2245
180 DATA 201,0,176,9,32,28,194,76,250,192,
   76,0,192,76,171,#1673
190 DATA 193,234,169,13,32,12,225,76,84,
   192,32,12,225,165,79,#1743
200 DATA 201,1,240,5,169,1,76,55,193,169,
   0,133,79,32,28,#1382
210 DATA 194,76,250,192,234,165,79,201,0,
   240,11,177,100,32,12,#1963
220 DATA 225,32,28,194,76,250,192,177,100,
   56,233,128,170,152,72,#2085
230 DATA 160,0,169,160,133,255,169,158,
   133,254,234,224,0,240,30,#2319
240 DATA 177,254,201,128,176,11,192,255,
   208,2,230,255,234,200,76,#2599
250 DATA 99,193,234,202,192,255,208,2,230,
   255,234,200,76,99,193,#2672
260 DATA 234,24,177,254,201,128,176,14,32,
   12,225,192,255,208,2,#2134
270 DATA 230,255,234,200,76,134,193,234,
   24,233,127,32,12,225,104,#2313
280 DATA 168,32,28,194,76,250,192,234,32,
   12,225,32,28,194,76,#1773
290 DATA 250,192,234,160,0,234,162,0,169,
```

```
    Ø,133,78,234,165,253,#2264
300 DATA 201,Ø,208,6,165,252,201,1Ø,144,
    3Ø,234,165,252,56,233,#2157
31Ø DATA 1Ø,133,252,165,253,233,Ø,133,253,
    224,255,24Ø,4,232,76,#2463
32Ø DATA 191,193,234,23Ø,78,162,Ø,76,191,
    193,234,165,252,1Ø5,48,#2352
33Ø DATA 72,165,78,133,253,2Ø1,Ø,2Ø8,11,
    138,2ØØ,2Ø1,1Ø,144,13,#1827
34Ø DATA 133,252,76,184,193,234,138,2ØØ,
    133,252,76,184,193,234,1Ø5,#2587
35Ø DATA 48,32,12,225,234,1Ø4,32,12,225,
    136,192,Ø,2Ø8,246,96,#18Ø2
36Ø DATA 234,192,255,2Ø8,2,23Ø,1Ø1,234,
    2ØØ,96,234,32,18,225,2Ø1,#2462
37Ø DATA 13,24Ø,6,149,85,232,76,38,194,234,
    2Ø2,134,96,96,76,#1871
38Ø DATA Ø,192,234,169,1,133,247,162,Ø,
    234,32,18,225,2Ø1,47,#1895
39Ø DATA 24Ø,6,149,85,232,76,67,194,234,
    2Ø2,134,96,162,Ø,234,#2111
4ØØ DATA 32,18,225,2Ø1,13,24Ø,6,149,89,
    232,76,87,194,234,2Ø2,#1998
41Ø DATA 228,96,24Ø,12,176,1Ø,169,32,234,
    232,149,89,228,96,2Ø8,#2199
42Ø DATA 248,234,134,83,16Ø,Ø,76,84,192,
    234,24,165,1Ø2,133,1Ø1,#197Ø
43Ø DATA 164,1Ø3,162,Ø,234,181,89,145,1ØØ,
    228,96,24Ø,1Ø,232,32,#2Ø16
44Ø DATA 28,194,76,137,194,76,57,194,234,
    162,Ø,32,28,194,177,#1783
45Ø DATA 1ØØ,213,85,24Ø,6,2Ø1,Ø,2Ø8,24Ø,
    234,96,234,165,1Ø1,133,#2256
46Ø DATA 1Ø2,132,1Ø3,234,232,32,28,194,
    177,1ØØ,213,85,2Ø8,22Ø,228,#2288
47Ø DATA 96,24Ø,187,76,182,194,76,57,194,
    169,Ø,133,251,169,16Ø,#2184
48Ø DATA 133,255,169,158,133,254,16Ø,Ø,
    162,Ø,177,254,213,85,24Ø,#2393
49Ø DATA 35,2Ø1,128,176,49,192,255,2Ø8,2,
    23Ø,255,2ØØ,177,254,2Ø1,#2563
5ØØ DATA 128,144,243,192,255,2Ø8,2,23Ø,
    255,2ØØ,23Ø,251,165,251,2Ø1,#2955
51Ø DATA 255,24Ø,199,76,216,194,192,255,
    2Ø8,2,23Ø,255,2ØØ,232,177,#2931
52Ø DATA 254,213,85,24Ø,242,2Ø1,128,144,
    2Ø7,233,128,213,85,2Ø8,214,#2795
53Ø DATA 165,251,24,1Ø5,128,133,251,162,Ø,
    149,85,169,Ø,76,82,#178Ø
54Ø DATA 192
6ØØ PRINT"□":M=49152:FOR Z=Ø TO
    53:C=Ø:FOR ZZ=Ø TO 14:READ X:POKE
    M,X
61Ø PRINT"□DATA LINE";PEEK(63)+PEEK
    (64)*256:C=C+X:M=M+1:NEXT ZZ:
    READ X$
62Ø IF VAL(RIGHT$(X$,LEN(X$)-1))<>C
    THEN PRINT "ERROR,CHECK LINE!":
    END
63Ø NEXT Z:READ X:POKE M,X:PRINT
    "□DATA OK."
```

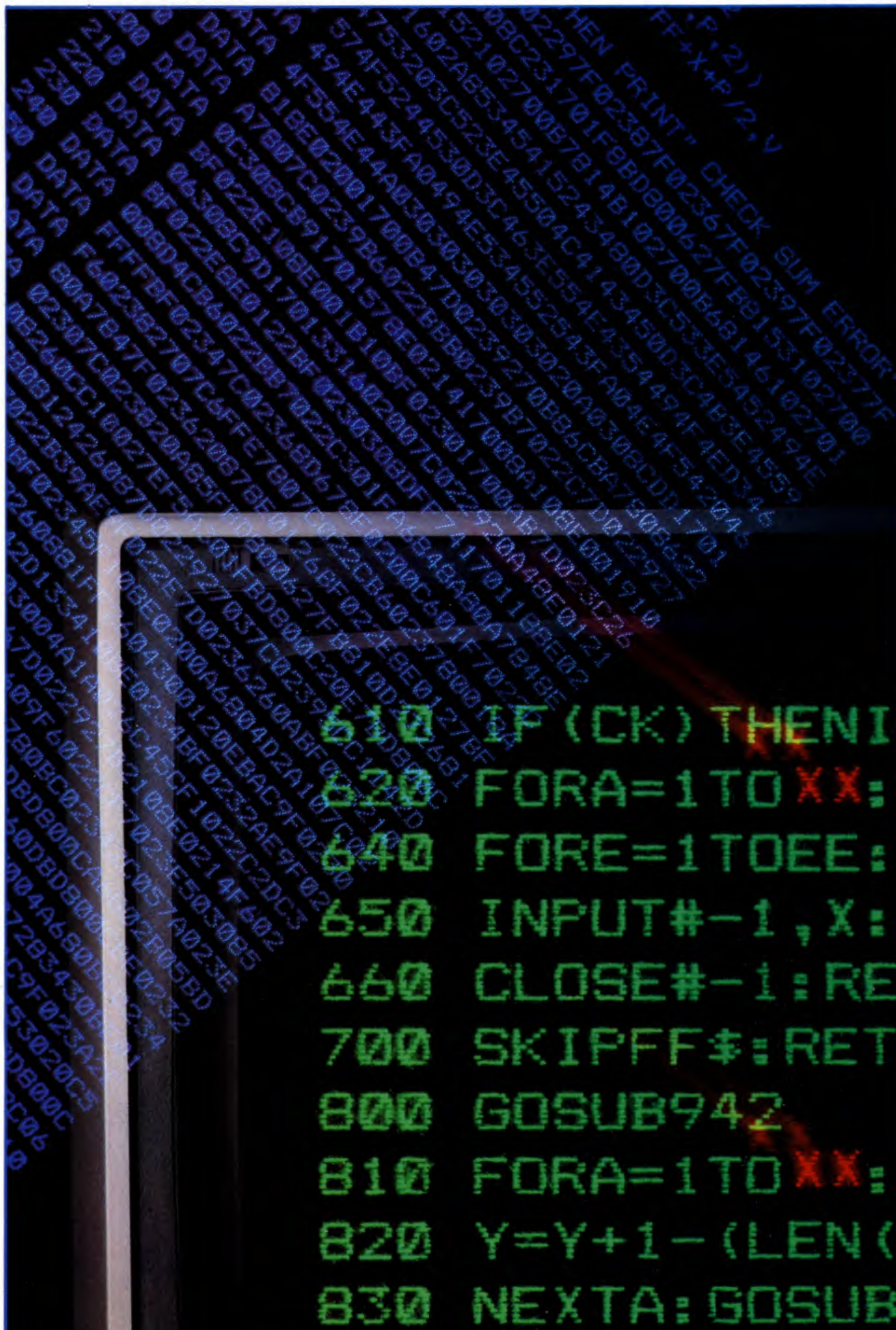
TROUBLE-SHOOTER

The Acorn program has been designed to work with both versions of BASIC, however, the machine code produced by RUNing the program will only work with the version of BASIC used to assemble it. So if you upgrade from BASIC I to BASIC II you'll need to create a new machine code program from the assembly version.

The reason for this is that the program uses tables of keywords that are in slightly different positions in the two versions of BASIC. When you RUN the assembly language it detects which version is in the computer and changes the value used for the start of the keyword table.

```
1Ø base%=&5ØØØ: HIMEM=base%
2Ø PROCSETUP
3Ø PROCASSEMBLE
4Ø PROCINFO
5Ø END
6Ø DEFPROCASSEMBLE
7Ø FORI%=ØTO3STEP3
8Ø P%=base%:[OPT I%
9Ø JMPgo:OPT□FNSS
1ØØ .go
11ØLDX#menl:LDY#menh:JSR prs
12Ø JSR get:CMP#88:BNE g1:RTS
13Ø .g1: CMP#83:BNE g2:JMP S
14Ø .g2: CMP#82:BNE g3:JMP R
15Ø .g3: CMP#75:BNE g4:JMP K
16Ø .g4: CMP#76:BNE g5:JMP L
17Ø .g5:CMP#67:BNE g6:JMP C
18Ø .g6: CMP#7Ø:BNE g7
19Ø LDA#&A4:STA S1:JMP K1
2ØØ .g7: CMP#8Ø:BNE go
21Ø LDA#&F2:STA S1:JMP K1
22Ø .S: LDX#insl:LDY#insh:JSR prs
23Ø LDA#1:JSR gets:BEQSX
24Ø JSR start
25Ø .SL: LDY#3:JSR sclns
26Ø BEQ S5:JSRpline
27Ø .S5: JSR gline:BNE SL
28Ø .SX:JMPgo
29Ø .R: LDX#insl:LDY#insh:JSR prs
3ØØ LDA#1:JSR gets:BEQ RX
31Ø LDX#inrl:LDY#inrh:JSR prs
32Ø LDA#2:JSR gets
33Ø JSR start
34Ø .RL: LDY#3
35Ø .RP4: JSR sclns:BNE RP8
36Ø JSR gline:BNE RL
37Ø .RX: JMP go
38Ø .RP8: STA SR
39Ø JSR pline:JSR repl:BNE RP6
4ØØ .RP9: JSR pline:JSR OSNWL
41Ø LDA SR:CLC:ADC L1:TAY:
    BNE RP4
42Ø .RP6
43Ø LDA L2:SEC:SBC L1:STA R4L:
    BEQ RR1
44Ø LDA#Ø:SBC#Ø:STA R4H
45Ø LDY#2:LDA(LL),Y:CLC:ADC R4L
46Ø STA R2L:LDA#Ø:ADC R4H
47Ø JSR chk:BEQ RP9
48Ø LDA R2L:STA(LL),Y
49Ø LDA SR:CLC:ADC LL:STA R1L
5ØØ LDA LH:ADC#Ø:STA R1H
51Ø LDAR1L:CLC:ADC L1:STA R1L
52Ø LDA R1H:ADC#Ø:STA R1H
53Ø LDA R1L:CLC:ADC R4L:STA R2L
54Ø LDA R1H:ADC R4H:STA R2H
55Ø LDA&12:SEC:SBC R1L:STA R3L
56Ø LDA&13:SBC R1H:STA R3H
57Ø LDA&12:CLC:ADC R4L:STA&12
58Ø LDA&13:ADC R4H:STA&13
59Ø LDA R4H:BMI RR2:JSR mvdn:
    JMP RR1
6ØØ .RR2: JSR mvup
61Ø .RR1: LDY SR:LDX#Ø
62Ø .RR5: CPX L2:BEQ RR6
63Ø LDA S2,X:STA(LL),Y
64Ø INY:INX:BNE RR5
65Ø .RR6: JSR pline:JSR OSNWL
66Ø LDA SR:CLC:ADC L2:TAY:JMP RP4
67Ø .chk
68Ø BNE chkf:LDA R2L:CMP#25Ø:BCS chkf
69Ø LDA R4L:CLC:ADC&12:STA R1L
7ØØ LDA R4H:ADC&13
71Ø CMP#base%DIV256:BCC chko
72Ø BNE chkf
73Ø LDA R4L:CMP#base%MOD256:
    BCS chkf
74Ø .chko: LDA#1:RTS
75Ø .chkf: LDX#chkl:LDY#chkh:JSR prs
76Ø LDA#Ø:RTS
77Ø .K: LDX#kwrl:LDY#kwrh:JSR prs
78Ø LDA#1:JSR gets:BEQ KX
79Ø JSR ctoke:LDA S1:BEQ KX
8ØØ .K1:JSR start
81Ø .KL: LDY#3:JSR sclnt
82Ø BEQ K2:JSR pline
83Ø .K2: JSR gline:BNE KL
84Ø .KX: JMP go
85Ø .L: LDX#inll:LDY#inlh:JSR prs
86Ø LDA#1:JSR gets:BEQ LX
87Ø JSR clino:JSR start
88Ø .LP: LDY#3:JSR sclnn
89Ø BEQ L5:JSR pline
9ØØ .L5: JSR gline:BNE LP
91Ø .LX: JMP go
92Ø .C: LDX#inll:LDY#inlh:JSR prs
93Ø LDA#1:JSR gets:BEQ CX
94Ø JSR clino:LDX#2
```

```
950 .CP1: LDA S1,X:PHA:DEX:BPL CP1
960 LDX#ilrl:LDY#ilrh:JSR prs
970 LDA#1:JSR gets:BEQ CX
980 JSRclino:LDX#Ø
990 .CP2:LDA S1,X:STA S2,X
1000 PLA:STA S1,X:INX:CPX#3:BNE CP2
1010 JSR start
1020 .CP: LDY#3
1030 .CP4: JSR sclnn:BEQ C5:PHA
1040 JSR pline:JSR repl:BEQ CP6
1050 PLA:PHA:CLC:ADC LL:STA R1L
1060 LDA LH:ADC#Ø:STA R1H
1070 LDY#2:LDX#2
1080 .CP3: LDA S2,X:STA(R1L),Y
1090 DEX:DEY:BPL CP3
1100 .CP6: JSR pline:JSR OSNWL
1110 PLA:CLC:ADC#3:TAY:BNE CP4
1120 .C5: JSR gline:BNE CP
1130 .CX: JMP go
1140 .start
1150 LDA&18:STA LH:LDA#1:STA LL:RTS
1160 .pline
1170 LDY#Ø:LDA(LL),Y:STA R1H
1180 INY:LDA(LL),Y:STA R1L:JSRpri
1190 LDA#2:STA SP:LDA#32:JSR
   OSWRITE
1200 .pll: INC SP:LDY SP:LDA(LL),Y
1210 CMP#13:BEQ plx
1220 CMP#34:BEQ plq
1230 CMP#141:BNE plnn:JSR dlino:JMPpll
1240 .plnn: JSR dtoke:JMP pll
1250 .plx: JSR OSNWL:RTS
1260 .plq: JSR OSWRITE
1270 INC SP:LDY SP:LDA(LL),Y
1280 CMP#13:BEQ plx
1290 CMP#34:BNE plq:JSR OSWRITE
1300 JMP pll
1310 .gline
1320 LDY#2:LDA(LL),Y:CLC:ADC LL:STA LL
1330 LDA#Ø:ADC LH:STA LH:LDY#Ø
1340 LDA(LL),Y:CMP#255:RTS
1350 .clino
1360 JSR d2b
1370 LDA R4L:AND#63:ORA#64:STA
   S1+1
1380 LDA R4H:AND#63:ORA#64:STA
   S1+2
1390 LDA R4L:AND#192:STA SG
1400 LDA R4H:AND#64:CLC
1410 RORA:ROR A:ORA SG
1420 ROR A:ROR A:EOR#84
1430 STA S1:RTS
1440 .dlino
1450 INC SP:LDY SP:LDA(LL),Y
1460 ASLA:ASL A:STA SG:AND#&CØ
1470 INY:EOR(LL),Y:STA R1L
1480 LDA SG:ASL A:ASL A:INY:EOR(LL),Y
1490 STA R1H:INC SP:INC SP:JSR pri
1500 RTS
1510 .ctoke
1520 LDA#tabs□DIV256:STA R1H
1530 LDA#tabs□MOD256:STA R1L
1540 LDY#128:LDX#Ø
1550 .ctl: LDA(R1L),Y:CMP S1,X
1560 BNE ctnx:INX:INY
1570 LDA S1,X:CMP#13:BEQ ctf:BNE ctl
1580 .ctnx: LDY#128:LDX#Ø
1590 .ct3: INC R1L:BNE ct1:INC R1H:.ct1
1600 LDA R1H:CMP#tabe□DIV256:BCC ctz
1610 BNE ctxx
1620 LDA R1L:CMP#tabe□MOD256:BCS
   ctxx
1630 .ctz:LDA(R1L),Y:CMP#127:BCC ct3
1640 LDA#2:CLC:ADC R1L:STA R1L
1650 LDA#Ø:ADC R1H:STA R1H:BNE ctl
1660 .ctf: LDA(R1L),Y:CMP#127:BCC ctnx
1670 STA S1:RTS
1680 .ctxx: LDA#Ø:STA S1:RTS
1690 .dtoke
1700 CMP#128:BCC ntk:BEQ and
1710 TAX
1720 LDA#tabs□DIV256:STA R1H
1730 LDA#tabs□MOD256:STA R1L
1740 LDY#128
1750 .tkl: TXA:CMP(R1L),Y:BEQ tkc
1760 INCR1L:BNE td:INC R1H:.td
1770 LDA R1H:CMP#tabe□DIV256:BNE tkl
```



```
1780 LDA R1L:CMP #tabe□MOD256:
   BNE tkl
1790 BEQ fin
1800 .tkc: LDX #0
1810 .tkh: DEY:INX:LDA(R1L),Y:
   CMP #127
1820 BCC tkh
1830 INY:INY:DEX:DEX:TXA:CMP #9:
   BCS fin
1840 .tke: LDA(R1L),Y:JSR OSWRITE
1850 INY:DEX:BNE tke
1860 .fin: RTS
1870 .ntk: JSR OSWRITE:RTS
1880 .and: LDA #65:JSR OSWRITE:
   LDA #78
1890 JSR OSWRITE:LDA #68:JSR OSWRITE:
   RTS
1900 .mvdn
1910 LDA R1H:CLC:ADC R3H:STA R1H
1920 LDA R2H:CLC:ADC R3H:STA R2H
1930 LDY R3L:BEQ DP
1940 .D1: DEY:LDA(R1L),Y:STA(R2L),Y
1950 CPY #0:BNE D1
1960 .DP: LDX R3H:BEQ DX
1970 .D2: DEC R1H:DEC R2H
1980 .D3: DEY:LDA(R1L),Y:STA(R2L),Y
1990 CPY #0:BNE D3:DEX:BNE D2
2000 .DX: RTS
2010 .mvup
2020 LDY #0:LDX R3H:BEQ UP
2030 .UL: LDA(R1L),Y:STA(R2L),Y
2040 INY:BNE UL
2050 INC R1H:INC R2H:DEX:BNE UL
2060 .UP: LDX R3L:BEQ UX
2070 .U1: LDA(R1L),Y:STA(R2L),Y
2080 INY:DEX:BNE U1
2090 .UX: RTS
2100 .d2b
2110 LDA #0:STA R4L:STA R4H:
   STA SG:TAX
2120 .d2bl
2130 LDX SG:LDA S1,X
2140 CMP #13:BEQ d2bx
2150 CMP #32:BEQ d2b2
2160 LDA R4L:STA R1L:LDA R4H:
   STA R1H
2170 LDA #0:STA R2H:LDA #10:
   STA R2L
2180 JSR mul:LDX SG
2190 LDA S1,X:SEC:SBC #ASC"0"
2200 CLC:ADC R3L:STA R4L
2210 LDA #0:ADC R3H:STA R4H
2220 .d2b2: INC SG:BNE d2bl
2230 .d2bx: RTS
2240 .mul
2250 LDA #0:STA R3L:STA R3H:
   LDX #16
2260 .ML: CLC:ROR R2H:ROR R2L
2270 BCS MA
2280 .MC:CLC:ROL R1L:ROL R1H
2290 DEX:BNE ML
```



```
2300 .MX: RTS
2310 .MA: LDA R1L:CLC:ADC R3L:
   STA R3L
2320 LDA R1H:ADC R3H:STA R3H
2330 JMP MC
2340 .pri
2350 LDA R1L:STA R2L
2360 LDA R1H:STA R2H
2370 LDA #0:STA SG
2380 LDA #16:STA R3L
2390 LDA #39:STA R3H
2400 JSR PS
2410 LDA #232:STA R3L
2420 LDA #3:STA R3H
2430 JSR PS
2440 LDA #100:STA R3L
2450 LDA #0:STA R3H
2460 JSR PS
2470 LDA #10:STA R3L
2480 JSR PS
2490 LDA #1:STA R3L:STA SG
2500 JSR PS
2510 RTS
2520 .PS: LDX #0
2530 .PP: LDA R1L:SEC:SBC R3L:
   STA R1L
2540 LDA R1H:SBC R3H:STA R1H
2550 BMI PE
2560 STA R2H:LDA R1L:STA R2L
2570 INX:JMP PP
2580 .PE: LDA R2L:STA R1L
2590 LDA R2H:STA R1H
2600 TXA:BNE PR:LDA SG:BEQ PX
2610 .PR:TXA:CLC:ADC #ASC"0"
2620 JSR OSWRITE
2630 LDA #1:STA SG
2640 .PX: RTS
2650 .sclns
2660 LDX #0
2670 .scl: LDA(LL),Y:CMP #141:BNE scc
2680 TYA:CLC:ADC #4:TAY:LDX #0:
   BEQ scl
2690 .scc: CMP S1,X:BNE scn
2700 INX:LDA S1,X:CMP #13:
   BEQ scf
2710 INY:BNE scl
2720 .scn: LDA(LL),Y:CMP #13:BEQ scfx
2730 TXA:BEQ scmx:STA SG:TYA
2740 SEC:SBC SG:TAY
2750 .scmx: INY:LDX #0:BEQ scl
2760 .scf: DEX:STX SG:TYA:SEC:SBC SG
2770 RTS
2780 .scfx: LDA #0:RTS
2790 .sclnn
2800 .nsl2: LDA(LL),Y:CMP #141:BNE nsl1
2810 LDX #0:INY
2820 .nsl: LDA(LL),Y:CMP S1,X:BNE nsf
2830 INY:INX:CPX #3:BNE nsl
2840 TYA:SEC:SBC #3:RTS
2850 .nsf: STX SG:LDA #3:SEC:SBC SG
2860 TYA:CLC:ADC SG
2870 TAY:BNE nsl2
2880 .nsl1: CMP #13:BEQ nsx:INY:
   BNE nsl2
2890 .nsx: LDA #0:RTS
2900 .sclnt: LDX #0
2910 .tsl: TXA:BNE tsq
2920 LDA(LL),Y:CMP #141:BNE tsc
2930 TYA:CLC:ADC #4:TAY:BNE tsl
2940 .tsc: CMP S1:BEQ tsf
2950 CMP #13:BEQ tsfa
2960 CMP #34:BNE tsn
2970 INX:.tsn: INY:BNE tsl
2980 .tsq: LDA(LL),Y:CMP #13:BEQ tsfa
2990 CMP #34:BNE tsq1:DEX
3000 .tsq1: INY:BNE tsl
3010 .tsf: TYA:RTS
3020 .tsfa: LDA #0:RTS
3030 .prs: STX R1L:STY R1H:LDY #0
3040 .psp: LDA(R1L),Y:CMP #13:
   BEQ psx
3050 CMP #124:BEQ psnl:JSR OSWRITE
3060 INY:BNE psp
3070 .psx: RTS
3080 .psnl:JSR OSNWL:INY:BNE psp
3090 .get: LDA #21:LDX #0:
   JSR OSBYTE
3100 LDA #62:JSR OSWRITE:
   JSR OSWRITE
3110 .gp: JSR OSREAD:BCS ge
3120 TAX:JSR OSWRITE:JSR OSNWL:
   TXA:RTS
3130 .ge: CMP #27:BNE gp
```

```
3140 LDA#126:JSR OSBYTE:BNE gp
3150 .gets: STA R1L:CMP#2:BEQ s2
3160 LDA#S1□MOD256:STA pblk
3170 LDA#S1□DIV256:STA pblk+1:BNE s1
3180 .s2: LDA#S2□MOD256:STA pblk
3190 LDA#S2□DIV256:STA pblk+1
3200 .s1: LDX#pblk□MOD256:LDY#pblk□DIV256
3210 LDA#0:JSR OSWORD:BCC gsx
3220 PLA:PLA:LDA#126:JSR OSBYTE
3230 JSR OSNWL:JMP go
3240 .gsx:LDA R1L:CMP#2:BEQ sx2
3250 STY L1:JMP sx1
3260 .sx2:STY L2
3270 .sx1: JSR OSNWL:TYA:RTS
3280 .repl: LDX#resl:LDY#resh:JSR prs
3290 JSR get:TAY:JSR OSNWL:TYA
3300 CMP#89:BEQ reply
3310 CMP#78:BNE repl:LDA#0:RTS
3320 .reply: LDA#1:RTS
3330 .end
3340 ]:NEXT
3350 ENDPROC
3360 DEFFNSS
3370 pblk=P%:P%=P%+5
3380 pblk?2=49:pblk?3=31:pblk?4=128
3390 S1=P%:S2=P%+50:P%=P%+100
3400 menl=FNE("¦¦< CROSS REFERENCER MENU >¦¦<S> Search for a string¦<R> Replace a string¦<K> Search for a keyword¦<L> Search for a line number¦<C> Change a line number¦<F> List functions¦<P> List procedures¦<X> Exit program¦¦"):menh=FNF
3410 insl=FNE("enter search string >"):insh=FNF
3420 inrl=FNE("enter new string >"):inrh=FNF
3430 inll=FNE("enter search line number >"):inlh=FNF
3440 ilrl=FNE("enter new line number >"):ilrh=FNF
3450 resl=FNE("¦replace ? Y/N "):resh=FNF
3460 kwrl=FNE("enter key word >"):kwrh=FNF
3470 chkl=FNE("¦not enough room !¦"):chkh=FNF
3480 =I%
3490 DEFFNE(A$)
3500 S%=P%:$P%=A$:P%=P%+LEN A$+1
3510 =S%MOD256
3520 DEFFNF:=S%DIV256
3530 DEFPROCSETUP
3540 LL=&70:LH=&71
3550 R1L=&74:R1H=&75:R2L=&76:R2H=&77
3560 R3L=&78:R3H=&79:R4L=&7A:R4H=&7B
3570 SP=&82:SG=&83:L1=&84:L2=&85:SR=&86
3580 OSWRITE=&FFEE:OSBYTE=&FFF4
3590 OSWORD=&FFF1:OSNWL=&FFE7
3600 OSREAD=&FFE0
3610 IF?&8015=50 tabs=&7FF1□ELSE tabs=&7FED
3620 tabe=&82ED
3630 ENDPROC
3640 DEFPROCINFO
3650 PRINT''"□Cross Referencer assembled at &";~base%
3660 PRINT"□To SAVE enter *SAVE CREF□";~base%;"□";~end+1
3670 PRINT"□To Use enter CALL&";~base%''
3680 ENDPROC
```

Tandy users will need to make some changes to the program. The numbers to change are all printed in bold. Change all occurrences of **8006** to A1C1, **800C** to A282 and **A0EA** to 8CC6. The checksums in these lines must also be changed. Change the last four digits in the following lines to: Line 110, 1657; Line 280, 1708; Line 290, 1923; Line 390, 1828; Line 400, 1790; Line 420, 1742; Line 440, 1961; Line 450, 1693; Line 470, 2003.

```
10 CLS: CLEAR 200,&H6FFF
20 FOR X=1 TO 741 STEP 18
30 READ X$: CS=0
40 FOR P=1 TO 36 STEP 2
50 V=VAL("&H"+MID$(X$,P,2))
60 CS=CS+V:POKE &H6FFF+X+P/2,V
70 NEXT: READA:IFCS<>A THEN PRINT "CHECKSUM ERROR IN LINE";100+10*INT(X/18):END
80 NEXT
100 DATA 34367F02297F02387F02367F02397F02377F,1141
110 DATA 023C308C231701F8BD800627FB8153102700,1437
120 DATA 668152102700B7814B102700B68146102701,1247
130 DATA 031602A85345415243480D3C533E5452494E,1168
140 DATA 4753203C523E45504C4143450D3C4B3E4559,1184
150 DATA 574F5244530D3C463E554E4354494F4ED346,1429
160 DATA 494E443FA0494E534552543FA04E4F542046,1477
170 DATA 4F554E44A030303030303020A0308CDD1701,1383
180 DATA 818E0200170 0B47D0239270B86CBA7808622,1510
190 DATA A7807C0239B6022BBB0239B7022C7D022927,1387
200 DATA 0C308CB91701578E021417008A108E001910,1020
210 DATA BF022E108E001B10BF02301700AB7D023C26,1100
220 DATA 06308C9D17013316020 07C022920A48E0121,989
230 DATA BF022E8E0122BF0230308DFF741701188E02,1409
240 DATA 008D4CB6022BB7022C301FA6848A80A7848E,1757
250 DATA FFFFBF02347C02368D678E0200C601F7022C,1815
260 DATA F602382707C6FFE7807C022CB6023780018A,1838
270 DATA 80A7847F023620878E0126BF022E8E0127BF,1570
280 DATA 02307C023820A85FBD800627FB810D272681,1488
290 DATA 08260CC10027EF5A301FBD800C20E7BD800C, 1619
300 DATA A7808124260 87D022927037C02395CC1142D,1249
310 DATA D1F7022B39AE9F022E7D0236260ABF023210,1427
320 DATA AE8410BF02345F108E0200A6804D2A107C02,1377
330 DATA 377D0238260881FF26043001 20EBAC9F0230,1407
340 DATA 2C5FBC02342D133410BE0234BF0232AE9F02,1335
350 DATA 32BF023435103004A1A026C45CF1022C2DC3,1590
360 DATA 7D023626388D467D022927B2108E0214F602,1299
370 DATA 2BF7023E7D0239260 9F6022CF7023E503085,1449
380 DATA A6A0E684C122270CA780BC02342C057A023E,1738
390 DATA 26EC128D1220813986 0DBD800CA6802B05BD,1676
400 DATA 800C20F73934367C023C860DBD800CBE0232,1486
410 DATA 8D71308DFE4F8DDCBE02323004A680BC0234,1967
420 DATA 2C414D2B05BD800C20F181FF27283430BE01,1590
430 DATA 21108E012210BF023A8D2EA680AC9F023A2E,1411
440 DATA 0D4D2B05BD800C20F0847FBD800C353020C5,1657
450 DATA A6803430BE0126108E012720D4860DBD800C,1541
460 DATA 3536391E89C0805D270EA680AC9F02302C06,1522
470 DATA 4D2AF55A26F239BD80062703BDA0EA342010,1839
480 DATA 8E303010BF708310BF708510BF70873002AE,1818
490 DATA 843001318DFDC23121301F8C00002711A6A4,1505
500 DATA 4CA7A4813A2DEA8630A7A4313F20EF352039,1911
510 DATA 353639,164
```

CLIFFHANGER: SNAKES ALIVE!

So far the snakes have just been tongue in cheek snakes in the grass. Now it is time to give them a bit of bite and add a vein of vitality to your vacillating vipers

Cliffhanger is now an all-singing all-dancing game. Everything moves, runs, jumps, bounces and flaps—except for the snakes. They've just been lying there, not doing much. Now's the time to give them a little hiss.

The following routine gives your snakes the shakes:

```
    org 59823
snk ld a,(57344)
    cp 2
    jr nc,sko
    ret
sko ld ix,5735Ø
    ld hl,425
    call skm
    ld hl,369
    call skm
    ld hl,282
    call skm
    ret
skm push hl
    ld de,(57354)
    sbc hl,de
    pop hl
    jr c,sns
    ret
sns ld a,(ix+Ø)
    inc a
    res 4,a
    ld (ix+Ø),a
    inc ix
    cp 7
    jr nc,sco
    ret
sco ld bc,57224
    ld d,43
    cp 15
    jr nz,ste
    ld bc,15616
    ld d,45
ste ld a,d
    call 58217
    ret
```

First of all you have to check to see whether you need any snakes. So the level number is loaded up from 57,344. This is compared with 2.

If you are on level three or four—that is, if the level number is set to 2 or 3—you need snakes. A compare is an unrecorded subtraction. So if you compare 2 with a number less than two, there is a borrow—so the carry flag is set. In that case the processor returns because snakes are not required.

If the carry flag isn't set, snakes are required so the **jr nc,sko** instruction jumps over the **ret**.

SNAKES ALIVE!

Memory location 57,35Ø carries the first of the so-called snake delays which tells the snakes whether their tongues should be out or not. IX is loaded with its address. HL is then loaded with 425, the position of the first snake's tongue. The **skm** routine which does the flicking is then called.

The next snake tongue position is loaded into HL and **skm** is called again. Then the third snake tongue position is loaded into HL and **skm** is called yet again.

And once all the tongues are flicking the routine returns.

FOR HEAVEN'S SNAKES

When the **skm** routine is called, the snakes' tongues' positions are pushed onto the stack for temporary storage. Then the sea's position is loaded up from 57,354 into DE. At this point you need to check whether the snakes have been drowned by the rising tide and gone to heaven—in which case they do not need their tongues flicked and this part of the routine can be skipped.

So the sea position in DE is subtracted from the tongue position in HL. If there is not a carry—that is, the sea position is above the snake's tongue position on the screen—the **jr c,sns** instruction does not operate and the processor returns. But if there is a carry and the sea position number is higher than the tongue position number, the **jr c,sns** jumps the processor over the **ret** and onto the rest of the routine.

TO FLICK OR NOT TO FLICK

The snake delay is loaded from the location pointed to by IX. The delay is then incremented and bit four is reset to stop the delay getting any greater than 15. The result is stored back in the location pointed to by IX.

IX is then incremented to move it onto the next snake delay.

The **cp 7** compares what is in the accumulator. And the **jr nc,sco** jumps the processor onto the routine that prints up the tongue, if the snake delay is 7 or more. If it is less than 7, the carry flag is set, the instruction does not make the jump and the processor returns.

So each time this routine is called a different snake delay is called out of locations 57,35Ø, 57,351 and 57,352—the pointer in IX is incremented between each call. And the snake's tongue flicks out for eight cycles and stays in for the next eight.

ONE TONGUE

Then, at last, you reach the routine that actually prints the snakes' tongues up on the screen. BC is loaded with 57,224, the data for the snake tongue. D is set to 43, the snake colour. Then the snake delay in A is compared to 15.

If it hasn't reached 15, the **jr nz** instruction takes the process on to **ste**. But if it has hit 15, you need to print a space over the tongue, so that the tongue will appear to be in for the next eight cycles. BC is loaded with 15,616, data for a blank space. And D is loaded with 45, the sky colour.

Whatever the value of the snake delay—as long as it is more than seven—the processor now reaches the instruction **ld a,d.** This loads the appropriate colour into A and the print routine at 58,2Ø8 is called.

This prints the snake's tongue—or a blank space—coming out of the appropriate snake's mouth. Then the processor returns.

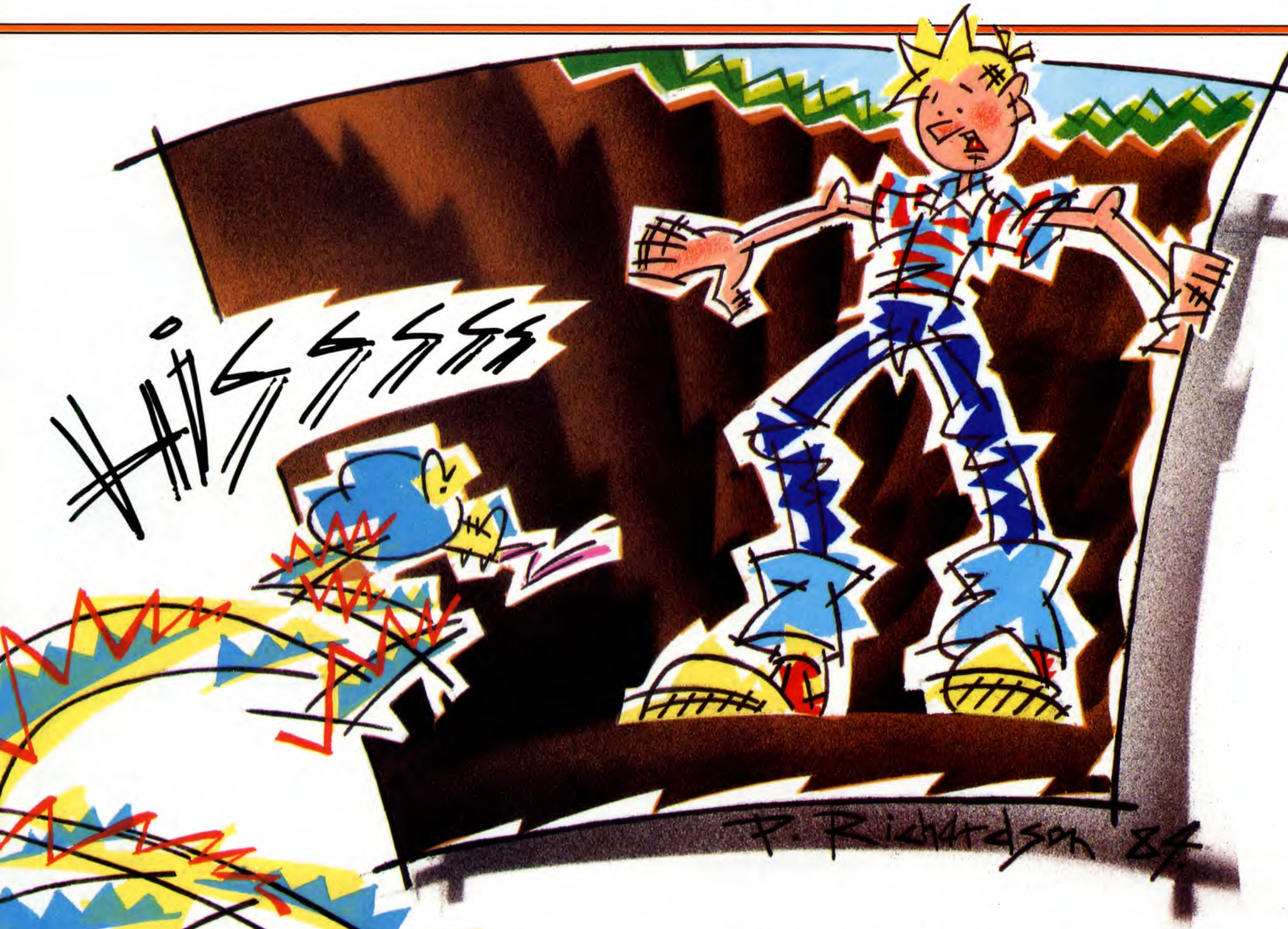
This routine flicks the snakes' tongues in and out:

```
ORG 22Ø16
LDA $CØØØ
CMP #1
BEQ FIN
CMP #2
BEQ FIN
CMP #3
BEQ FIN
```



- SNAKE DELAYS
- CHECKING FOR A FLICK
- FLICKING THE TONGUE
- MOVING FROM TONGUE TO TONGUE

The 'CLIFFHANGER' listings published in this magazine and subsequent parts bear absolutely no resemblance to, and are in no way associated with, the computer game called 'CLIFF HANGER' released for the Commodore 64 and published by New Generation Software Limited.




```
      LDX #Ø
      LDY #3
LOOP  DEC $C364,X
      BEQ IN
      LDA $C364,X
      CMP #1Ø
      BEQ OUT
RET   INX
      DEY
      BNE LOOP
FIN   RTS
IN    LDA #235
      STA $Ø75B,X
      LDA #2Ø
      STA $C364,X
      JMP RET
OUT   LDA #234
      STA $Ø75B,X
      JMP RET
```

The first thing this routine does is to check whether you are on a level that has snakes—you wouldn't want disembodied tongues flicking up and down on the screen.

So the level number is loaded up from memory location $CØØØ into the accumulator and compared to 1, 2 and 3. If any of these are found, the BEQ instruction following their CMPs takes the processor off to the RTS marked by the label FIN and it returns.

TWO TONGUED

The X register is used as an offset which points to the right snake and the Y register is used as a counter. X is loaded with Ø so it starts on the first and Y is loaded with 3, to count round the loops.

Then the first thing that happens when the processor goes into the tongue-flicking routines is that the so-called snake delay in memory location $C364 plus X is decremented. This is used to slow the tongue down so that it stays out for a period and in for a period—otherwise, you wouldn't see it.

Locations $C364, $C365 and $C366 contain a value between Ø and 2Ø. If the value is between Ø and 1Ø, the corresponding snake's tongue is in. If it is between 1Ø and 2Ø, the snake's tongue is out.

These values are set at the start of the program by the initialization routine.

Once the contents of the first snake delay have been decremented, the BEQ instruction checks to see whether it has reached zero. If it

has the processor branches off to the label OUT which marks the beginning of the routine that flicks the snake's tongue out.

If the snake delay has not decremented to zero, it is loaded into the accumulator and compared with 1Ø.

If it is 1Ø, the BEQ instruction takes the processor off to the label IN which marks the beginning of the routine that flicks the snake's tongue in.

But if the snake delay is neither Ø or 1Ø, the processor continues. It increments X—that is, it moves onto the next snake—and decrements Y.

If the contents of the counter in Y have not been decremented to Ø—and all three snakes have not been dealt with—the BNE instruction loops back and goes through the same process again, on the next snake.

But if it has counted down to zero, all the tongue work has been done and the processor proceeds. It hits the RTS following and returns.

TONGUE OUT

To get a snake to stick its tongue out, 235 is loaded into the accumulator and stored in the memory location $Ø75B plus X. Locations $Ø75B, $Ø75C and $Ø75D are the sprite pointers for the three snakes and 235 defines the location where the data for the snake with its tongue out is to be located (see page 172). The number 2Ø is then loaded into the accumulator and stored back in the appropriate snake delay, so it can start all over again next time the processor comes by.

The processor then jumps back to the label RET, increments X to move onto the next snakes, decrements the counter in Y, branches back if another snake has to be dealt with, or returns if it doesn't.

TONGUE IN

If the snake has his tongue in, 234 is loaded into the accumulator and stored in the appropriate sprite pointer. 234 is the data for the picture of the snake with its tongue in.

That done, the processor jumps back to the label RET again.

The following program moves the snakes' tongues in and out. Their movements are controlled by counters in memory locations &2162, &2163 and &2164. Each time the snake routine is called these counters are decreased by 1. When any of them reaches Ø, the corresponding snake's tongue is moved and the counter is reset to the value held in memory locations &2165 to &2167.

The actual routine that moves the snakes' tongues is held in locations &2168 to &21B8. The tongues are moved in and out by exclusively oring the colour red—so if the snake's tongue is out, it goes in and, if it's in, it goes out.

Don't forget to set the computer up as usual before you key this part of the Cliffhanger game in.

```
30 FORPASS=
   ØTO3STEP3
40 P%=&21A6
50 [OPTPASS
60 .Snake
70 LDX#3
80 LDY#Ø
90 .Lb1
100 DEC&219F,X
110 BNELb2
120 LDA&21A2,X
130 STA&219F,X
140 STX&70
150 STY&71
160 LDA#5
170 JSR&FFEE
180 LDA#18
190 JSR&FFEE
200 LDA#3
210 JSR&FFEE
220 LDA#1
230 JSR&FFEE
240 LDA&1928,Y
250 ASLA
260 TAX
270 LDA#33
280 SEC
290 SBC&1929,Y
300 ASLA
310 TAY
320 JSR&1964
330 LDA#238
340 JSR&FFEE
350 LDA#4
360 JSR&FFEE
370 JSR&14C8
380 LDX&70
390 LDY&71
400 .Lb2
410 INY
420 INY
430 DEX
440 BNELb1
450 RTS
460 ]NEXT
470 DATA1,7,13,20,30,25
480 FORA%=&21A0TO&
    21A5:READ?A%:NEXT
490 ?&14BF=16
```

To test this routine, SAVE it, LOAD the rest of the routines in from memory, then key in the commands:

```
CALL&1D77:?&83=4:CALL&1D9B:REPEAT:
CALL&21A6:FOR A%=ØTO100:NEXT:UNTILØ
```

TONGUE WAG

The instructions in Lines 7Ø and 9Ø set up a loop. Essentially, dealing with three snakes' tongues is doing the same thing three times over. So the X register is loaded with a 3, to count down, while the Y register is loaded with Ø, to count up.

Once inside the loop the snake counter given by &219F offset by the contents of X is decremented. The result is compared with zero. If it is not zero, the BNE instruction in Line 11Ø branches forward to the end of the routine where the loop counters are set for the next time round. So if this snake counter has not decremented to zero, the processor moves on to decrement and check the next one for zero.

But if the snake counter has decremented to zero, the processor moves on to the next instruction and loads up the value from &21A2 offset by X. This is stored in &219F



offset by X, restoring the value of the appropriate counter.

The values of X and Y are then stored in the zero-page memory locations &7Ø and &71 as various subroutines are called and their values may be corrupted.

TIP OF THE TONGUE

Next, values are output through the accumulator to the screen routine at &FFEE. The 5 in Line 16Ø gives a VDU5 which moves the text cursor to the same place as the graphics cursor.

The 18 in Line 18Ø gives a VDU18 or GCOL. And the subsequent 3 and 1, in Lines 2ØØ and 22Ø, gives a GCOL 3,1 which exclusively ors the colour red next time something is printed.

Next the cursor has to be positioned. The snakes' positions are held in &1928 and &1929, &193Ø and &1931, and &1932 and &1933. But the X and Y coordinates held in these locations are not in the proper form for the cursor moving routine, because this is the same one used to move the boulder and Willie, and uses *half* character squares. So when the X coordinate is loaded into the accumulator by the instruction in Line 24Ø, it is doubled by doing an arithmetic shift to the left. The result is then transferred into the X register by the transfer the contents of A into X instruction in Line 26Ø.

The Y coordinate does not work the same way round either. So 33 is loaded into the accumulator, the carry flag is set and the contents of &1929 offset by Y are taken away. Then the result is doubled by an arithmetic shift to the left and transferred into the Y



register by the instruction in Line 31Ø.

The processor then jumps to the subroutine at &1964 which moves the cursor to the position given by the X and Y registers.

GIVE IT A HISS

The number 238—the UDG number of the snakes' tongue—is loaded into the accumulator and output to the screen through the subroutine at &FFEE. This prints the tongue on the screen in the cursor position. But as a GCOL 3,1 instruction is in force, if a tongue is already on the screen in that position, it is overprinted by a tongue in the background colour. So if there's no tongue there, a red one is printed, but if a red one is there, it is blanked out by overprinting it with a blue one.

Once that has been done, a 4 is loaded into the accumulator and output through the subroutine at &FFEE. This separates the graphics and text cursors again.

Then the processor jumps to the subroutine at &14C8 which gives the snakes' hissing sound. The values of X and Y are then restored by loading those registers again with the values stored in &7Ø and &71.

Y is incremented twice to move it along to the next pair of snake coordinates. And X is decremented as it is counting the loops.

If it has not counted down to zero, the BNE instruction in Line 44Ø branches the processor back to deal with the next snake's tongue. But if it has reached zero, all three snakes' tongues have been dealt with and the branch is not made. In that case the processor proceeds to the next instruction—an RTS—and returns.

Following that there is a little bit of DATA. This is READ into &21AØ to &21A5 to give the initial and restore values for the snakes' tongues. They are staggered so that all the tongues do not flick in unison, or at the same speed.

The following routine gives your snakes the shakes:

```
       ORG 2Ø856
SNK    LDA 18238
       CMPA #2
       BHS SKO
       RTS
SKO    LDY #18255
       LDX #5Ø95
       JSR SKM
       LDX #4591
       JSR SKM
       LDX #3833
       JSR SKM
       RTS
SKM    PSHS X
       LDX 18247
       LEAX 31,X
       CMPX ,S
       PULS X
       BHI SNS
       RTS
SNS    LDA ,Y
       INCA
       ANDA #$F
       STA ,Y+
       CMPA #7
       BHS SCO
       RTS
SCO    LDU #18Ø62
       CMPA #15
       BNE STE
       LDU #1536
STE    LEAX -256,X
       JSR CHARPR
       RTS
CHARPR EQU 194Ø2
```

First of all you have to check to see whether you need any snakes. So the level number is loaded up from 18,238 and compared to 2.

If you are on level three or four—that is, if the level number is set to 2 or 3—you need snakes. And the BHS—Branch if Higher or Same—instruction skips the processor over the RTS following into the main routine if snakes are required.

MAKING THE HISS

Y is loaded with the address of the first snake delay. This is the variable that stops the snake shaking too fast. The second and third snake delays follow directly after it in memory.

X is then loaded with 5Ø95, the screen position of the first hole. And the processor jumps to the SKM subroutine given lower down in the program.

The SKM routine shakes the first snake and increments the contents of the Y register before it returns. So when the processor jumps to it a second and third time—to shake the second and third snakes—only the second and third hole's screen positions have to be loaded into the X register. Y is incremented automatically along the three snake delays.

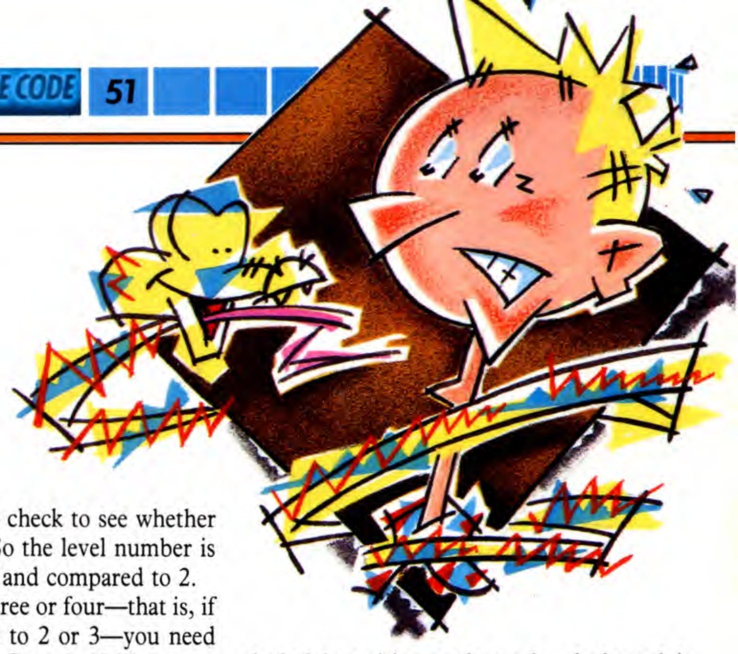
And when all three snakes have been dealt with, the processor returns.

RATTLING THE RATTLER

The first thing the processor does when entering the SKM routine is to push the snake's screen position in X onto the stack for temporary storage.

The X register is then loaded with the contents of 18,247, the position of the sea. It is then incremented with 31 to move the screen pointer to the extreme right-hand side of the screen. Then X is compared to the position of the snake's pit, which is on the hardware stack, by the instruction CMPX ,S. The snake's position is then pulled off the stack again into X.

Pulling does not effect any of the flags though. So the BHI—Branch if HIgher—instruction still operates on the CMPX,S. So if the sea is higher than the hole—that is, the sea's screen position in X is not higher than



the hole's position on the stack—the branch is not made and the processor returns.

But if the sea is not higher than the hole—that is, the sea's screen position in X is higher than the hole's position on the stack—the processor skips over the RTS and continues with the snake-shaking routine.

VACILLATE A VIPER

A is loaded with the contents of the snake delay pointed to by the contents of the Y register. The snake delay is then incremented and ANDed with $F to take it back to zero again every time it is incremented over 15.

The result is stored back in the snake delay pointed to by Y and Y is incremented to move it onto the next snake delay.

The snake delay which is still in A is compared with 7 and the processor branches on to the snake-tongue printing routine if it is 7 or over. Otherwise the processor returns.

VIPER WIPER

When the processor enters the SCO routine it loads the user stack pointer with 18,Ø62, the address of the data for the snake's tongue.

A—which still contains the incremented snake delay—is compared to 15. If it hasn't been incremented up to 15, the processor branches forward to the print instructions. But if it has reached 15, U is loaded with 1536, the address of the plain sky in the top left-hand corner of the screen.

Either way, X is decremented by 256. This moves the screen pointer up to one character square above the snake's hole. This is the position of the tongue.

If the snake delay was between 7 and 14, the tongue is printed in that position when the CHARPR subroutine is called. But if the snake delay was 15, plain sky is printed there to blank out the old tongue.

The tongue remains blanked out until the snake delay is incremented up to 7 again.

CUMULATIVE INDEX

An interim index will be published each week. There will be a complete index in the last issue of *INPUT*.

COMING IN ISSUE 49...

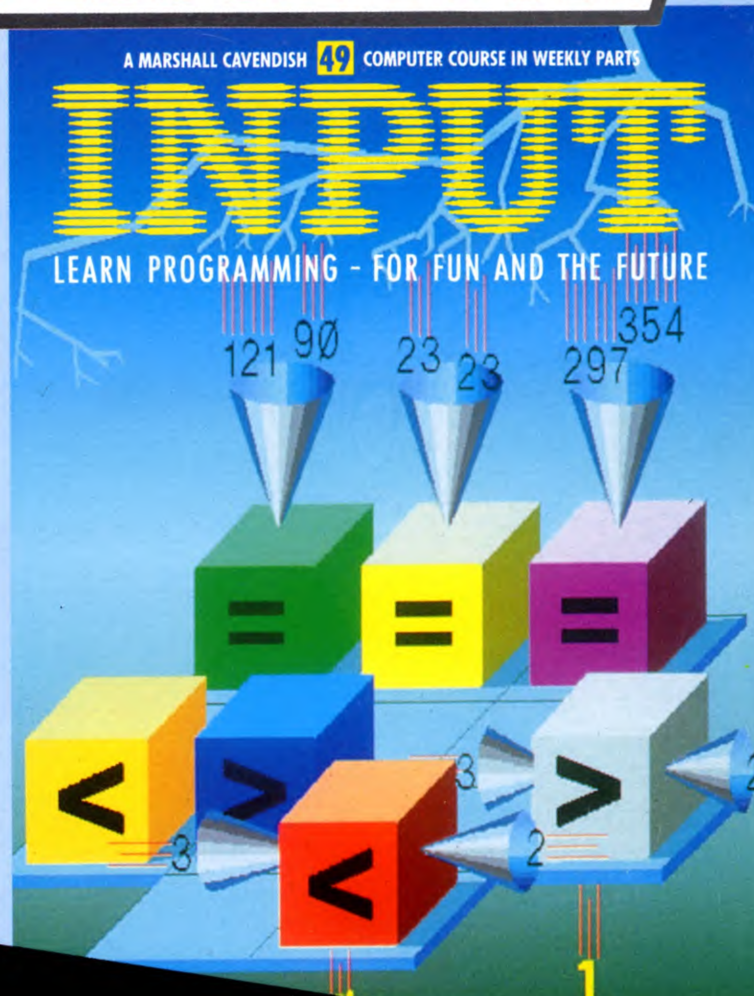
❑ *The microelectronics inside your computer can be used to control all sorts of external devices. Discover an unlimited field for experiments when you put COMPUTERS IN CONTROL*

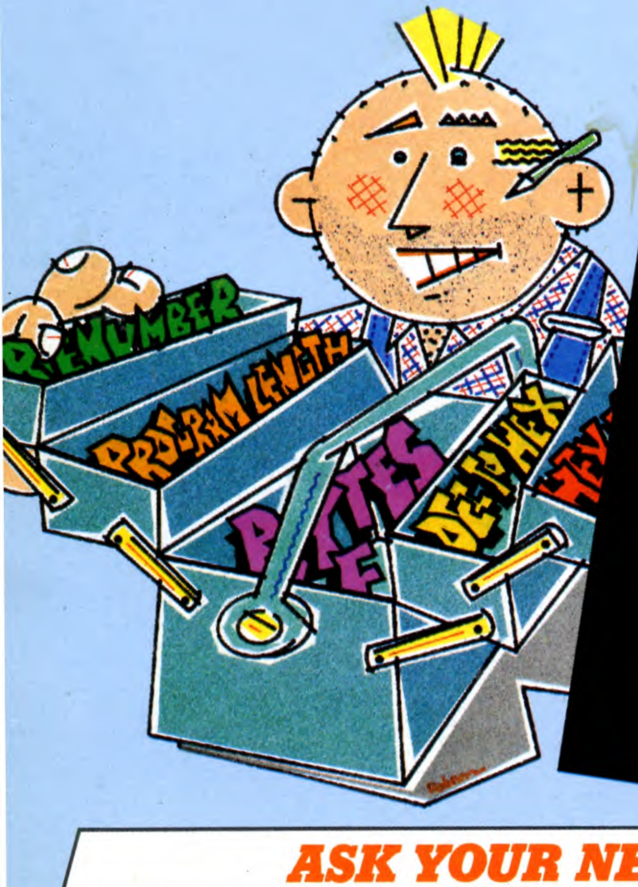
❑ *CLIFFHANGER comes to life. In this part you'll add the final routine, the MAIN ACTION ROUTINE that forms the heart of the complete game*

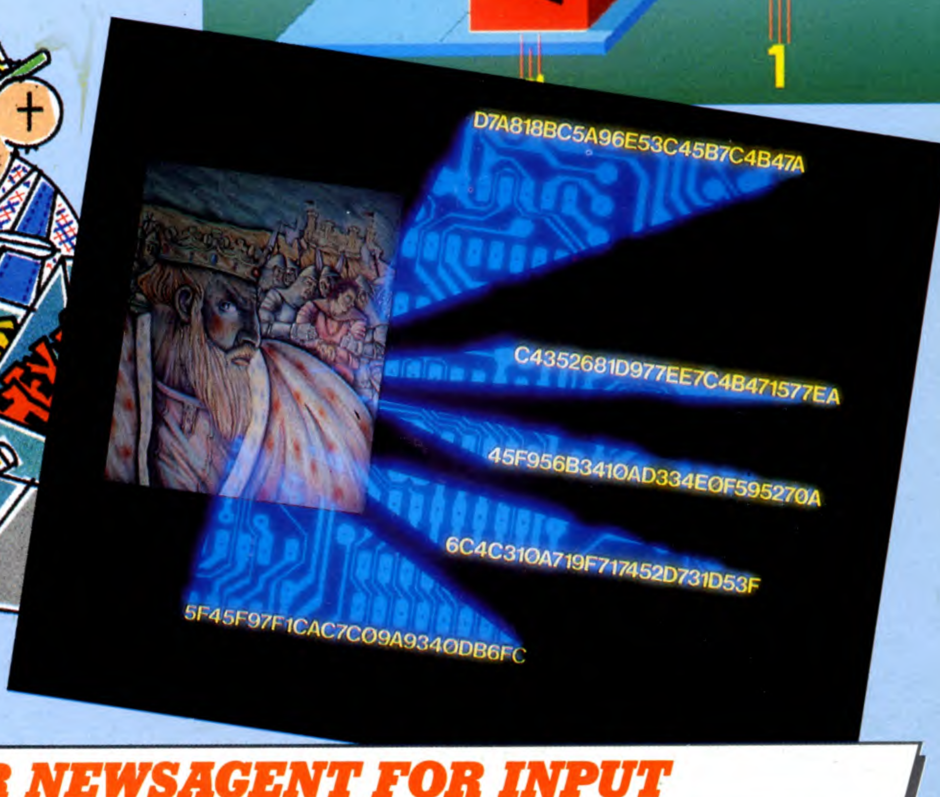
❑ *The last article on FORTH looks at the routines you can use to build up complex PROGRAM STRUCTURES*

❑ *The ADVENTURE GAME is nearly ready to run, but still needs the CODED MESSAGES—start entering them*

❑ *And for SPECTRUM and COMMODORE users, there's a useful aid to program development in the shape of a MACHINE CODE TOOLKIT*







ASK YOUR NEWSAGENT FOR INPUT